UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"

Pós-Graduação em Ciência da Computação

Juliana Cassiano Ferrarezi

LIBVIEWS - UMA FERRAMENTA WEB PARA VISUALIZAÇÃO DE BIBLIOTECAS E SUAS DEPENDÊNCIAS EM SISTEMAS DE INFORMAÇÃO

São José do Rio Preto
2017

Juliana Cassiano Ferrarezi

LIBVIEWS - UMA FERRAMENTA WEB PARA VISUALIZAÇÃO DE BIBLIOTECAS E SUAS DEPENDÊNCIAS EM SISTEMAS DE INFORMAÇÃO

Orientador: Prof. Dr. José Remo Ferreira Brega

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Processamento de Imagens e Visão Computacional, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

São José do Rio Preto
2017

Ferrarezi, Juliana Cassiano.
Libviews : uma ferramenta web para visualização de bibliotecas e suas dependências em sistemas de informação / Juliana Cassiano Ferrarezi. -- São José do Rio Preto, 2017
82 f. : il., tabs.

Orientador: José Remo Ferreira Brega
Dissertação (mestrado) – Universidade Estadual Paulista “Júlio de Mesquita Filho”, Instituto de Biociências, Letras e Ciências Exatas

1. Computação. 2. Engenharia de software. 3. Administração de projetos. 4. Software - Desenvolvimento. 5. Visualização da informação. I. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. II. Título.

CDU – 518.72

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Juliana Cassiano Ferrarezi

LIBVIEWS - UMA FERRAMENTA WEB PARA VISUALIZAÇÃO DE BIBLIOTECAS E SUAS DEPENDÊNCIAS EM SISTEMAS DE INFORMAÇÃO

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Comissão Examinadora

Profº Drº José Remo Ferreira Brega
UNESP - Bauru
Orientador

Profº Drº João Pedro Albino
UNESP - Bauru

Profº Drº Robson Augusto Siscoutto
Unoeste – Presidente Prudente

São José do Rio Preto
30 de janeiro de 2017

Agradecimentos

Primeramente agradeço a Deus por me permitir realizar mais esse sonho.

Aos meus pais José Luiz Ferrarezi e Maria Aparecida Cassiano Ferrarezi por sempre me incentivarem e apoiarem em tudo que foi necessário para chegar até aqui.

Ao meu marido, por estar ao meu lado e acompanhar essa caminhada.

Ao Professor Remo (José Remo Ferreira Brega), não apenas como professor mas também como amigo, por toda aprendizagem profissional e pessoal. Também agradeço a equipe de trabalho do Laboratório de Interfaces e Visualização (LIV da UNESP Campus Bauru), especialmente ao Mário Popolin Neto e ao Alessandro Campanhã de Moraes, por toda ajuda.

Aos professores que participaram das bancas e contribuíram com o desenvolvimento deste trabalho.

Ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) da UNESP, pela oportunidade.

Resumo

Bibliotecas de software são importantes e comumente usadas por permitir que os desenvolvedores utilizem funções básicas já implementadas e se concentrem em atividades complexas relacionadas diretamente às regras de negócio do software em desenvolvimento. Além do que, a disponibilização de bibliotecas na internet facilita sua utilização em larga escala. No entanto, pode haver problemas no desenvolvimento de software quanto a utilização de várias bibliotecas desenvolvidas por terceiros, uma vez que são projetos independentes que funcionarão em conjunto. Este trabalho apresenta o LibViews, um software que, por meio de técnicas de Visualização da Informação, disponibiliza uma representação gráfica de projetos de software e as bibliotecas que eles utilizam. A ferramenta apresentada também possibilita a análise de cada biblioteca utilizada através de métricas que permitem analisar a evolução de bibliotecas. O LibViews foi desenvolvido para proporcionar uma melhor compreensão das bibliotecas e suas versões, bem como a utilização de bibliotecas em projetos de software. Dessa forma, o software permite o entendimento das dependências do software, ou seja, bibliotecas de terceiros utilizadas que interferem diretamente no funcionamento do software. Pode-se afirmar, portanto, que o LibViews ajuda no planejamento, desenvolvimento e manutenção de projetos, permitindo a descoberta de informações até então desconhecidas. Como exemplo, o LibViews foi aplicado em um projeto de software administrativo de uma universidade, comprovando os benefícios de sua utilização para compreender a relação entre o projeto de software e suas dependências.

Palavras-chave: Visualização da informação, Bibliotecas de software, Dependências de software, Gerenciamento de dependência de software

Abstract

Software libraries are important and commonly used for allowing developers to use basic functions already implemented and to focus on complex activities directly related to the business rules of the software being developed. In addition, the availability of libraries on the Internet facilitates their mass use. However, there may be problems in software development regarding the use of various libraries developed by third parties, since they are independent projects that will work together. This work presents the LibViews, a software for visualization of software projects and their dependencies; And analysis of each library used through metrics that allow to analyze the evolution of libraries. LibViews was developed to provide a better understanding of libraries and their versions, as well as the use of libraries in software projects. Thus, the software allows the understanding of the software's dependencies, that is, third-party libraries used that interfere directly in the operation of the software. LibViews, therefore, can assist in the planning, development, and maintenance of projects, allowing the discovery of previously unknown information. As a use case, LibViews has been applied in a university administrative software project, proving the gains from its use to understand the relationship between the software project and its dependencies.

Keywords: Information visualization. Software libraries. Software dependencies. Software dependency management.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

API	*Application Programming Interface*
CEM	*the amount of Change in Existing Methods*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
1D	Uma dimensão
2D	Duas dimensões
3D	Três dimensões
JAR	*Java ARchive*
JDBC	*Java Database Connectivity*
JSF	*JavaServer Faces*
JSON	*JavaScript Object Notation*
LOC	*Lines of Code*
McCabe	Métricas de complexidade
NRM	*Number of Removed methods*
NNM	*Number of New Methods*
NSM	*Number of Snapshot Methods*
PNM	*the Percentage of New Methods*
POM	*Project Object Model*
RCNO	*the Ratio of Change in New to Old methods*
SVG	*Scalable Vector Graphics*
VI	Visualização da Informação
XML	*eXtensible Markup Language*
W3C	*tWorld Wide Web Consortium*
WRM	*the Weighted Number of removed methods*

Sumário

1 Introdução

Este Capítulo apresenta as considerações iniciais desta dissertação, bem como a motivação, os objetivos do trabalho e sua organização.

1.1 Considerações Iniciais

Bibliotecas são componentes de software, muitas vezes desenvolvidos por terceiros, que disponibilizam funções prontas através de interfaces. Essas funções podem ser referentes a validações, layout e comunicação com banco de dados, por exemplo.

Também conhecidas como *frameworks*, as bibliotecas podem ser desenvolvidas em diversas linguagens e serem acopladas aos sistemas de informação de diferentes formas a fim de atender as necessidades comuns. Como exemplo, é possível citar bibliotecas escritas em C++, JavaScript, PHP, Java, entre outras.

Com o uso de bibliotecas como dependências (ou seja, um software depende de bibliotecas de terceiros para seu funcionamento) os desenvolvedores de software não precisam se preocupar com funcionalidades comuns à grande maioria dos sistemas de informação, mas sim com funcionalidades referentes ao sistema que está sendo produzido no momento.

Porém, como todo sistema de informação, uma biblioteca pode ter seu desenvolvimento estendido e muitas versões da mesma podem ser geradas, para corrigir problemas ou mesmo incrementar o que já existe. Nesses casos, pode ocorrer incompatibilidades entre as dependências de um projeto que usa uma versão antiga com a nova versão.

Para entender essas alterações que geram incompatibilidade pode ser utilizada a Visualização da Informação (VI), que é composta por técnicas que visam apresentar conjuntos de dados por meio de representações visuais de forma que o entendimento dos dados em análise seja facilitado. Considerando a grande quantidade de informações a que os usuários tem acesso atualmente, essas técnicas auxiliam a análise dos dados, facilitando a identicação de informações relevantes e possibilitando a descoberta de informações antes desconhecidas.

1.2 Motivação

A disseminação do uso de bibliotecas no desenvolvimento de software e a complexidade das informações sobre o relacionamento entre um sistema de informação e as bibliotecas que ele utiliza impulsionaram as pesquisas desta dissertação.

1.3 Objetivo

O objetivo principal desse trabalho é apresentar uma aplicação para auxiliar o entendimento dos projetos de software quanto às bibliotecas que utilizam, por meio de técnicas de Visualização da Informação.

A fim de alcançar o objetivo principal, foram traçadas metas:

- Estudar trabalhos relacionados;
- Desenvolver uma aplicação com uma representação gráfica dos dados;
- Testar a ferramenta com usuários para verificar a importância do conhecimento que a representação disponibiliza e a satisfação do usuário ao utilizar a aplicação.

1.4 Organização

Esta dissertação está organizada nos capítulos descritos abaixo, além deste Capítulo 1, que apresenta as considerações iniciais, a motivação e o objetivo:

- O Capítulo 2, **Bibliotecas e Dependências**, apresenta conceitos desta área e exemplos de frameworks;
- O Capítulo 3, **Visualização da Informação**, apresenta conceitos desta área, tipos de dados e suas representações;
- O Capítulo 4, **Revisão Sistemática**, apresenta um estudo de trabalhos científicos relacionados a bibliotecas e dependências e Visualização da Informação, bem como trabalhos relacionados;
- O Capítulo 5, **LibViews**, apresenta o aplicativo para visualização dos dados de bibliotecas e dependências de software;
- O Capítulo 6, **Resultados e Discussão**, apresenta pesquisa de satisfação feita após a utilização da aplicação por usuários; e
- O Capítulo 7, **Conclusão**, apresenta as considerações finais sobre o trabalho.

2 Bibliotecas e Dependências

Neste capítulo estão apresentados conceitos de bibliotecas de sofwtare e suas dependências em um sistema de informação, além de exemplos de *frameworks* utilizados em diferentes linguagens de programação.

2.1 Conceitos

Segundo Teyton et al. (2013), uma biblioteca é um componente de software que oferece funcionalidades facilmente invocadas através de interfaces bem definidas. O autor também destaca que o amplo uso de bibliotecas de terceiros é quase obrigatório nos sistemas de software modernos.

As bibliotecas também podem ser chamadas de artefatos e, para Bauer (2013), a reutilização de artefatos no desenvolvimento de sistemas tornou-se uma estratégia utilizada em larga escala, com a finalidade de entregar sistemas ricos em recursos e um bom custo-benefício.

O uso de bibliotecas, segundo Dietrich, Jezek e Brada (2014), é facilitado pela combinação de fatores sociais como a disseminação de comunidades de sistemas com códigos abertos e produtos comerciais.

Um exemplo de utilização é a biblioteca do Hibernate que é amplamente usada pelos profissinais que trabalham com a linguagem Java (DIETRICH; JEZEK; BRADA, 2014) e será melhor apresentada na seção 2.2.1.

Outro exemplo é dado pela utilização de bibliotecas por desenvolvedores que trabalham em aplicações para web como calendários, mapas, redes sociais, entre outras funcionalidades (KEIL; THIEMANN, 2013). Na seção 2.2.2 será apresentada uma biblioteca bastante utilizada para esse tipo de aplicação, o jQuery.

A reutilização de bibliotecas de terceiros promete aumentar significativamente a produtividade no desenvolvimento de sistemas. Entretanto, depender de bibliotecas externas e outras APIs trazem riscos para o projeto e impacto nas tomadas de decisões durante seu desenvolvimento e manutenção (BAUER; HEINEMANN; DEISSENBOECK, 2012). A razão é que *frameworks* (como também são chamadas as bibliotecas que facilitam atividades básicas de um sistema) estão sendo muito utilizados e isso significa sistemas independentes que estão sendo desenvolvidos ao mesmo tempo, até mesmo por equipes diferentes, e utilizados em conjunto (HUMPHREY et al., 2014).

2.2 Exemplos de bibliotecas de software

Nesta seção serão descritos exemplos de bibliotecas que facilitam atividades comuns a vários sistemas de informação, como conexão com banco de dados e tela de apresentação.

2.2.1 Hibernate

Hibernate é um framework para persistência de dados baseado na linguagem de programação Java. Além de fornecer o mapeamento de tabelas do banco de dados para classes Java, ele permite consulta, modificação e mecanismos de recuperação de dados. Esta biblioteca oferece técnicas de otimização de desempenho dos sistemas e um recurso que deve ser destacado é a técnica de cache (JING; FAN, 2011). Para isso, o Hibernate possui uma arquitetura definida que está apresentada na Figura 1.

Segundo Wu e Yin (2010), utilizando o Hibernate, os programadores Java podem usar a programação pensando para manipular objetos de banco de dados como quiser e o framework pode ser aplicado a qualquer tarefa que utiliza JDBC.

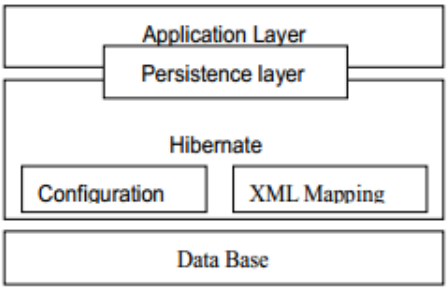

Figura 1 – Arquitetura do Hibernate.

Fonte: Xue e Zhu (2009).

Conforme descrito em Hibernate (2014), basta fazer o download do arquivo JAR da versão escolhida do framework e adicioná-lo como dependência, para que todos seus recursos fiquem disponíveis no sistema. Na Figura 2 está apresentado um exemplo de código que utiliza recursos do Hibernate.

```
/**
 * Alterar Senha
 *
 * @param senhaAtual
 * @param novaSenha
 * @throws NamingException
 */
public void alterarSenha(Usuario usuario, String senhaAtual, String novaSenha) throws ServiceValidationException, NamingException {
    usuario.setSenha(Crypt.crypt(novaSenha));
    save(usuario);

    log.info(ConfigHelper.getProperty("info.saveOk", "Alteração de Senha"));
}
```

Figura 2 – Exemplo da facilidade para inserção de dados no JDBC.

Fonte: Da autora.

2.2.2 jQuery

jQuery é uma biblioteca JavaScript que simplifica a criação de páginas web dinâmicas, ajuda na manipulação de eventos e tem prontas várias tarefas que são executadas no cliente e não no servidor. Diminuindo o tráfego entre cliente e servidor as funcionalidades ficam mais rápidas (DHAND, 2012).

Para Li e Li (2013), jQuery é uma biblioteca JavaScript rápida, leve e rica em recursos, que pode ser baixada do endereço *http://jquery.com*. Para sua utilização, basta a inclusão dela no projeto e utilização nas páginas HTML, como os exemplos apresentados nas Figuras 3 e 4.

```
1  <%@page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
2  <%@taglib uri="http://displaytag.sf.net" prefix="display" %>
3  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4  <%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
6  <html>
7  <head>
8      <script type="text/javascript" src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.7.min.js">
9  </script>
10 </head>
```

Figura 3 – Exemplo de inclusão do jQuery.

Fonte: Da autora.

```
jQuery.ajax({
    url: strUrl,
    data: arrData,
    type: 'POST',
    dataType: 'xml',
    contentType: "application/x-www-form-urlencoded;charset=UTF-8",
    success: function(xml) {
        $('#txt_superior').empty();
        $('#txt_superior').append( new Option(' - ', 0) );
        $(xml).find('entry').each(function(key, value){
            var id = $(value).attr('key');
            var text = $(value).text();
            if(id == idSuperiorSeleciona){
                $('#txt_superior').append( new Option(text, id, true, true) );
            }else{
                $('#txt_superior').append( new Option(text, id) );
            }
        });
    }
});
```

Figura 4 – Exemplo de utilização do jQuery.

Fonte: Da autora.

Exemplos de utilização de jQuery citados em Pierro et al. (2010):

- Criação de alertas para os usuários, como por exemplo: quando o usuário insere um expressão errada no campo de consulta;
- Manipulação de objetos HTML, como abrir novas janelas e enviar formulários; e
- Interface de tabelas, das quais os dados podem ser ordenados, pesquisados e filtrados, como apresentado na Figura 5.

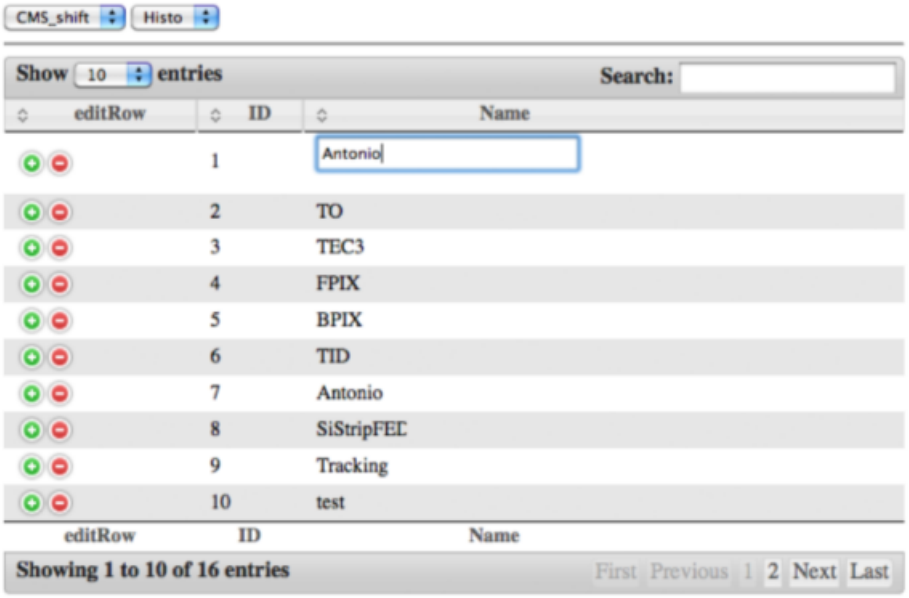

Figura 5 – Exemplo de tabela gerada utilizando jQuery.

Fonte: Pierro et al. (2010).

2.3 Considerações finais

As bibliotecas ou *frameworks* permitem que os desenvolvedores reutilizem funcionalidades já desenvolvidas por terceiros, como por exemplo a persistência de dados. Isso permite que sistemas mais complexos possam ser desenvolvidos com menor custo. Por essa razão, bibliotecas estão sendo utilizadas em larga escala.

Porém essa reutilização de bibliotecas de terceiros traz riscos ao projeto, por serem projetos independentes pode haver incompatibilidades ao trabalharem em conjunto.

3 Visualização da Informação

Neste capítulo estão apresentados conceitos de Visualização da Informação (VI), tipos de dados e suas representações, que serão utilizadas neste trabalho para auxiliar o entendimento de projetos de software e suas dependências.

3.1 Conceitos

A grande quantidade de informações é considerada um problema na interação humano-computador da atualidade. A internet disponibiliza milhões de documentos e nem todas as informações são relevantes. O computador pode, em poucos segundos, apresentar ao usuário informações que ele poderia levar anos para entender (CARR; UNIVERSITET, 1999).

A Visualização da Informação consiste em técnicas que aumentam a capacidade cognitiva humana por meio de representações gráficas dos dados. Assim, o sistema visual humano pode ser usado para auxiliar na identificação de padrões, tendências e valores discrepantes, permitindo uma melhor compreensão dos dados e a descoberta de novas informações e conhecimento (MORAES; ELER; BREGA, 2014).

Ou seja, a VI visa apresentar ao usuário as informações intuitivamente para um entendimento mais rápido do que é relevante (CARR; UNIVERSITET, 1999).

Técnicas de visualização podem permitir a exibição de grandes conjuntos de dados sem perder a capacidade de mostrar as informações mais relevantes (ALHENSHIRI; BLUSTEIN, 2010). Portanto, a medida de uma visualização efetiva pode ser a habilidade para gerar novas descobertas além das tarefas predefinidas de análise dos dados (SARAIYA; NORTH; DUCA, 2005).

Segundo Ware (2012), o processo de VI inclui quatro estágios básicos, como apresentado na Figura 6, são estes:

- Coleta e armazenamento dos dados;
- Pré-processamento responsável por transformar os dados a fim de facilitar sua manipulação;
- Mapeamento dos dados selecionados para uma representação visual, que é realizado através de algoritmos; e
- O sistema perceptivo e cognitivo humano.

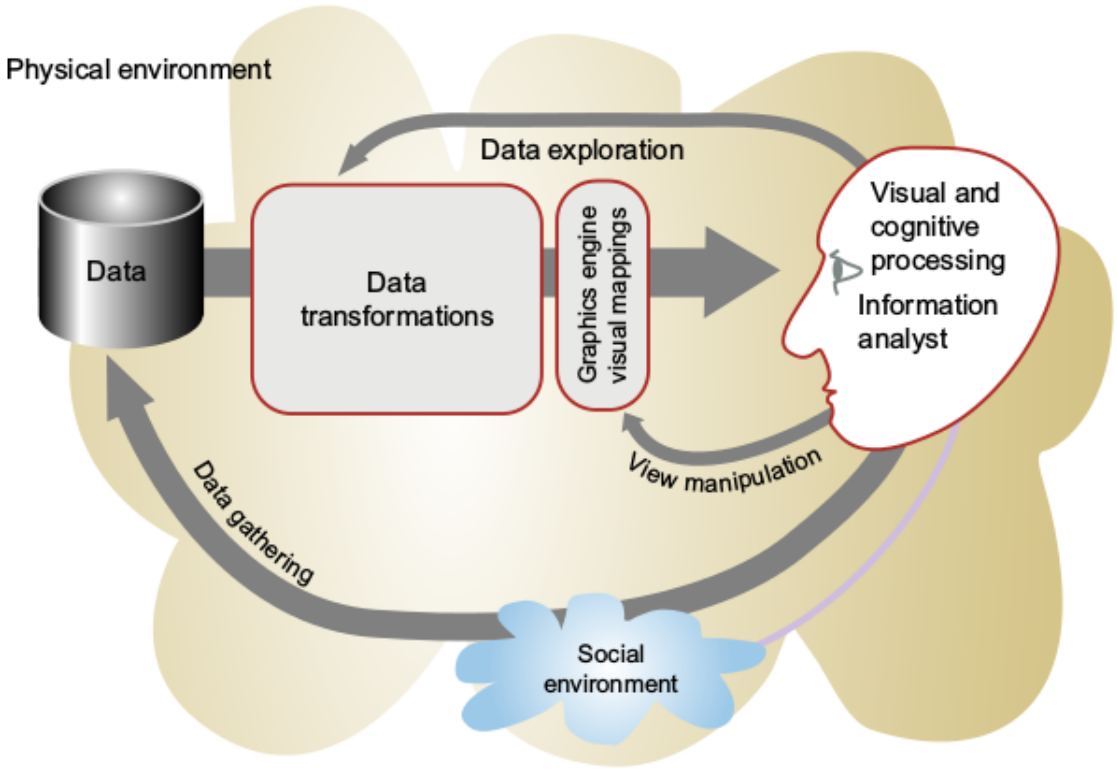

Figura 6 – Processo de VI.

Fonte: Ware (2012).

Segundo Andrews (2013) e como pode ser visto na Tabela 1 e na Figura 7, a VI é importante porque torna mais fácil identificar tendências e padrões, bem como fazer comparações na representação visual.

Figura 7 – Gráfico de vendas de 2012 (Valor X Vendedor).

Fonte: (ANDREWS, 2013).

A representação de dados é a forma utilizada para apresentar dados através de uma escolha de formas físicas; se é através de uma linha, uma barra, um círculo, ou em qualquer outra variável visual, toma-se os dados como matéria-prima e cria-se uma representação para melhor retratar os atributos (KIRK, 2012).

Tabela 1 – Tabela de vendas de 2012 (Valor X Vendedor).

Month	**Salesperson A**	**Salesperson B**
2012-01	28366	23274
2012-02	27050	21732
2012-03	29463	23845
2012-04	32561	28732
2012-05	28050	24023
2012-06	30100	26089
2012-07	22343	19026
2012-08	21506	17903
2012-09	24664	19387
2012-10	28842	23490
2012-11	30621	25873
2012-12	36254	28490

Fonte: (ANDREWS, 2013).

Para criar uma representação cria-se uma metáfora visual, ou seja, utiliza-se conceitos metafóricos, que são aqueles que são entendidos e estruturados não apenas em termos próprios, mas sim em termos de outros conceitos. Isso envolve conceituar um tipo de objeto ou experiência em termos de um tipo diferente de objeto ou experiência. As metáforas são comumente usadas como uma forma de entender como diferenças sutis na forma da linguagem podem sugerir diferentes interpretações da mesma informação (LAKOFF; JOHNSON, 1980).

Outras características visuais podem ser incluídos na visualização, como: cor, recursos interativos e anotações explicativas. O uso eficaz da cor fornece um atrativo, devido a natureza do olho e do cérebro humano. A capacidade de selecionar, filtrar, excluir ou modificar certas variáveis é uma forma valiosa de deixar o usuário interagir com diferentes fatias de dados. Além disso, agrupamento e classificação são opções comuns para o entendimento dos dados e descoberta de novas informações. Anotações explicativas fornecem significado extra, intuição e profundidade de conhecimento para os usuários (KIRK, 2012).

3.2 Tipos de dados e suas Representações

Se o objetivo da VI é melhorar a percepção visual dos dados, deve-se conhecer os tipos de dados a serem visualizados, para poder afirmar, por exemplo, que cores são boas para mercado de ações, e texturas são boas para mapas geológicos (WARE, 2012).

Segundo Andrews (2013), os tipos de dados são: lineares, hierarárquicos, redes, multidimensional e feature space.

Selecionar o método de visualização apropriado deve levar em conta as seguintes perguntas: como é que você vai mostrar e o que você quer dizer (KIRK, 2012).

Dentre esses tipos de dados, as dependências de um projeto de software (dados que serão apresentados nesse trabalho), são classificadas como Redes, ou, segundo Kirk (2012): conexões de plotagem e relacionamento, que é um tipo de dado que tem como objetivo de comunicação avaliar as associações, as distribuições e padrões que existem entre os conjuntos de dados multivariados. Esta coleção de soluções reflete alguma das mais complexas soluções visuais e geralmente se concentra em facilitar a análise exploratória.

Ainda segundo Kirk (2012) as possíveis representações para esse tipo de dados são:

Scatter Plot: apresentado na Figura 8, também é chamado de gráfico de dispersão. É uma combinação de duas variáveis quantitativas traçadas para os eixos x e y, a fim de revelar padrões de correlações, agrupamentos e valores discrepantes. A posição dos dados e cores auxiliam a representação dos dados, essas são chamadas de variáveis visuais.

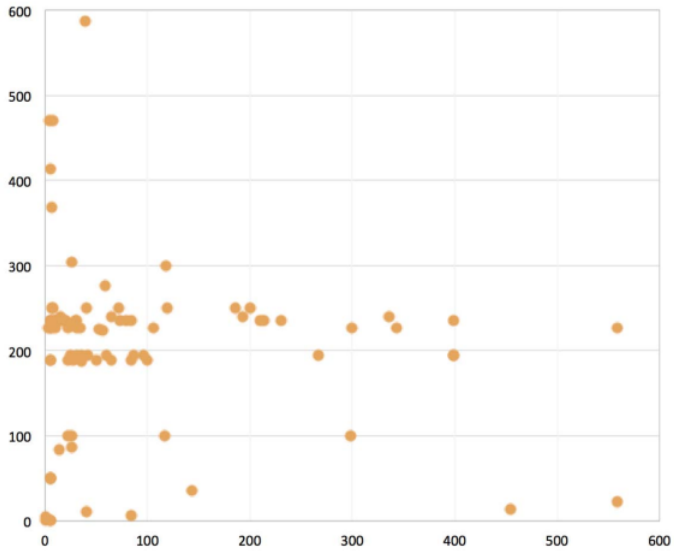

Figura 8 – Scatter Plot.

Fonte: (KIRK, 2012).

Bubble Plot: ou gráfico bolha, estende o potencial de um gráfico de dispersão através de múltiplas codificações dos dados. Além de variáveis quantitativas, permite representar variáveis categóricas, ou seja, categorias de dados diferentes. No exemplo da Figura 9, nota-se as marcas tornando-se círculos de tamanho variável e, em seguida, colorido de acordo com a sua relação categórica. As variavéis visuais são: posição, área e cor.

Scatter Plot Matrix: similar ao pequeno gráfico Scatter Plot está apresentado na Figura 10, além de duas variáveis quantitativas, o Scatter Plot Matrix, também representa categorias diferentes, através de cores. As variavéis visuais são: posição e cor.

Heatmap: com um exemplo apresentado na Figura 11, permite identificar a correspon-

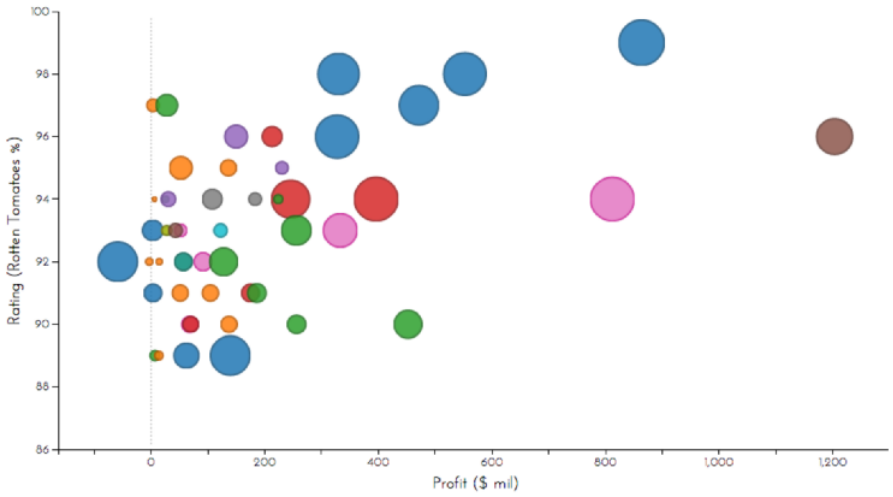

Figura 9 – Bubble Plot.

Fonte: (KIRK, 2012).

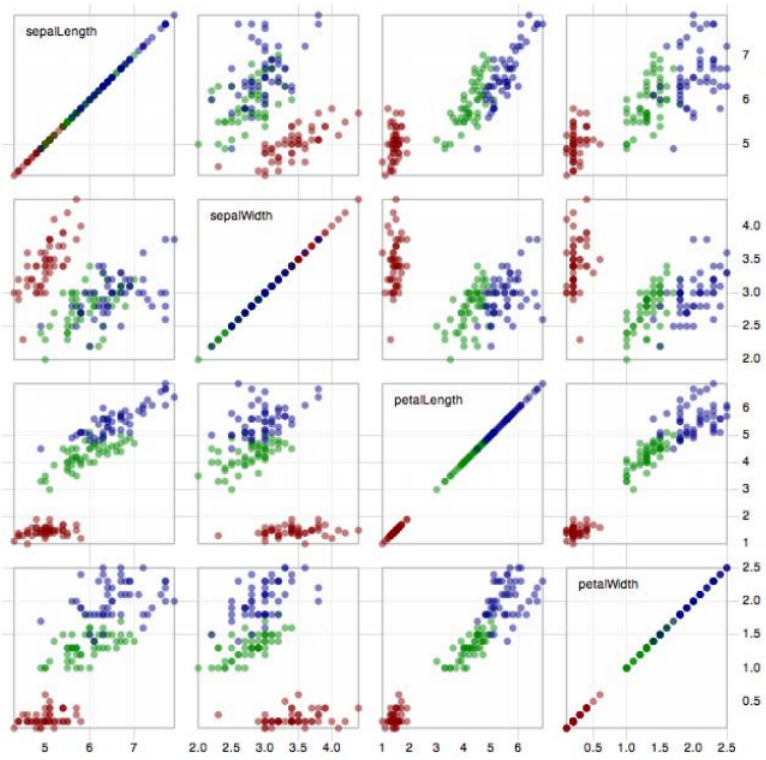

Figura 10 – Scatter Plot Matrix.

Fonte: (KIRK, 2012).

dência de padrões para detectar a ordem e a hierarquia de valores quantitativos através de uma matriz de combinações. A utilização de um esquema de cor com a diminuição da saturação ou o aumento da leveza ajuda a criar a sensação da classificação de magnitude dos dados. As variáveis visuais são: posição e cor.

Parallel Sets: apresentado na Figura 12, são conjuntos paralelos que oferecem uma maneira única de explorar e analisar visualmente o conjuntos de dados. A técnica consiste em

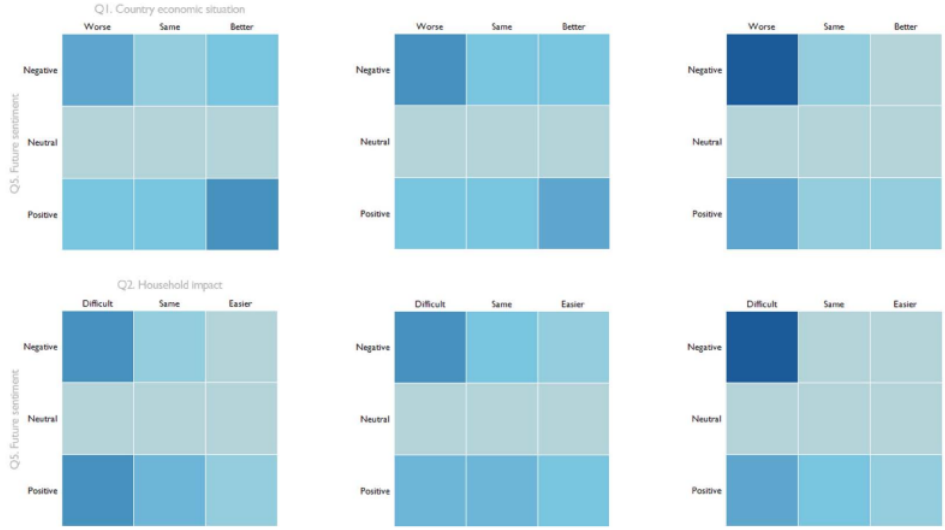

Figura 11 – Heatmap.

Fonte: (KIRK, 2012).

relacionar todos os seus dados a uma série de eixos, para uma cada uma das variáveis que será examinada, isto cria caminhos que mostra as conexões entre a desagregação dos valores contidos nos seus dados para cada variável. Eles são úteis para aprender sobre os potenciais correlações e consistências que existem nos bancos de dados. As variáveis visuais são: posição, largura, link e cor.

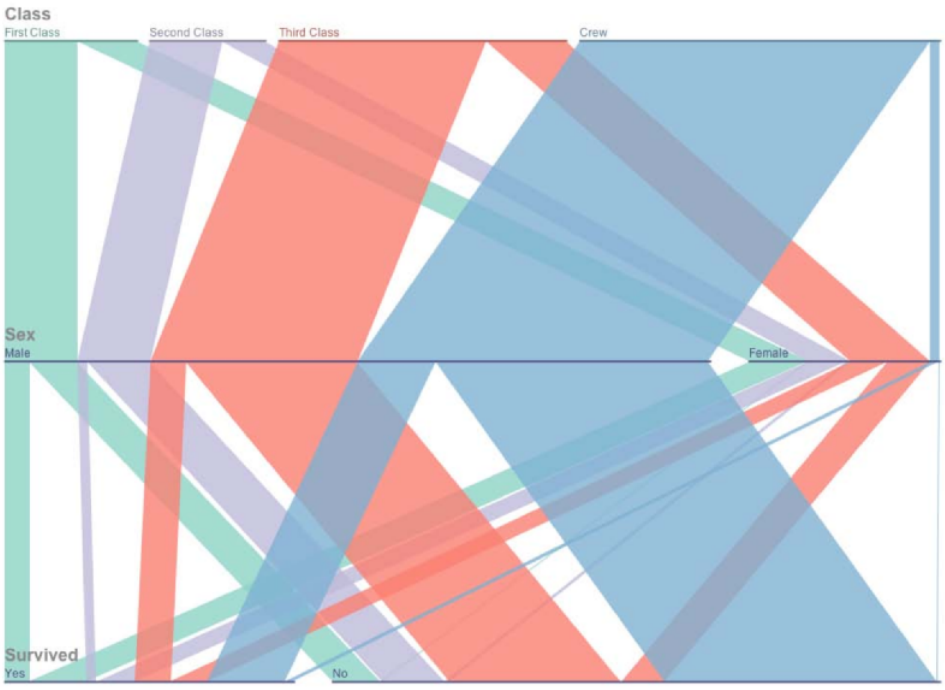

Figura 12 – Parallel Sets.

Fonte: (KIRK, 2012).

Rede Radial: uma rede radial, apresentada na Figura 13 cria um quadro para comparar complexas relações entre valores categóricos. O uso de representações radiais de layout dá a oportunidade de ir além das restrições de emparelhamento de eixos x e y. A chave da propriedade explicativa são as ligações que existem entre os componentes, por vezes, tamanho (espessura) e colorido para incorporar camadas extras de detalhes.

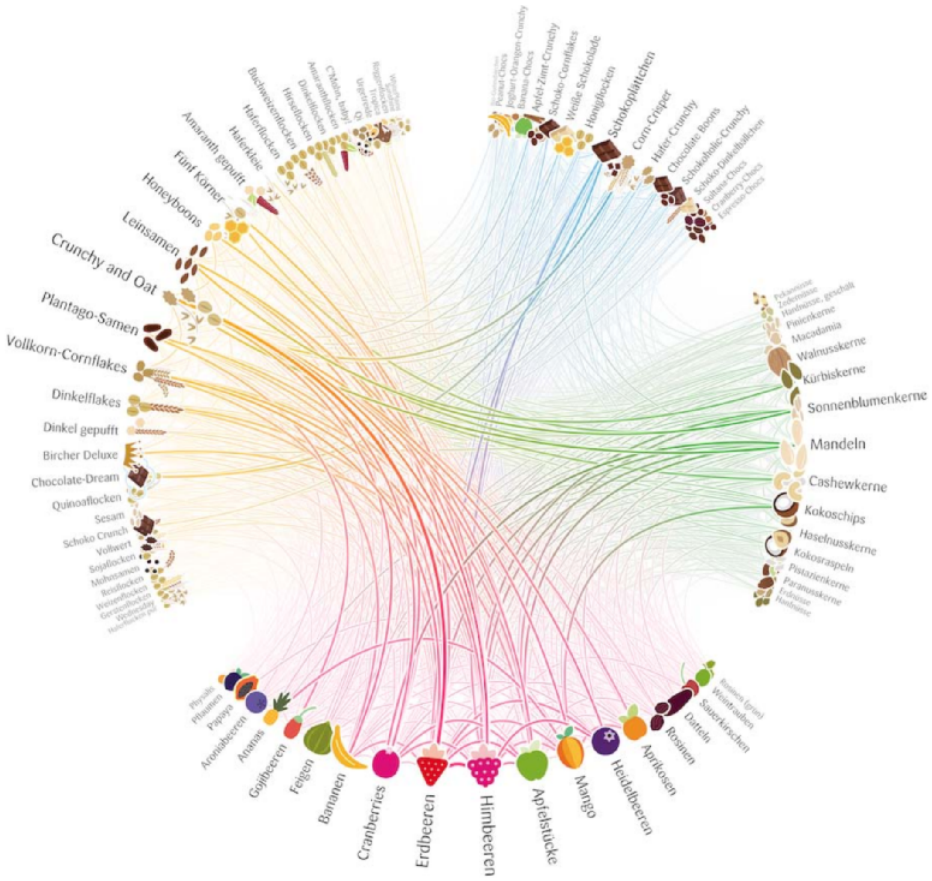

Figura 13 – Rede Radial.

Fonte: (KIRK, 2012).

Network Diagram: apresentado na Figura 14, facilita a exploração de estruturas complexas de dados com base na existência ou na força quantificável de relacionamentos e conexões.

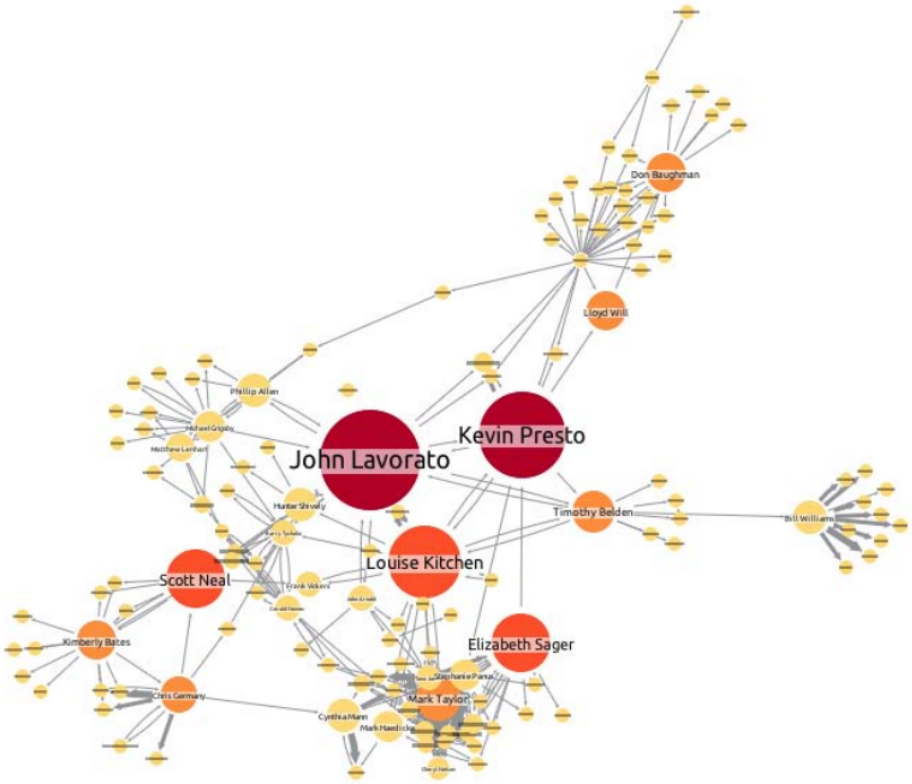

Figura 14 – Network Diagram.

Fonte: (KIRK, 2012).

3.3 Considerações finais

Representações gráficas podem melhorar significativamente a compreensão de dados, gerando descoberta de informações e auxiliando na tomada de decisões.

Como também visto neste capítulo, uma representação pode auxiliar na apresentação dos dados para que se tenha um entendimento rápido de grande quantidade de informações, como detalhes de um projeto de software e suas dependências.

Conhecendo e seguindo os estágios básicos da VI, foi construída a representação do capítulo 5.

4 Revisão Sistemática

Neste capítulo estão apresentados o conceito de revisão sistemática, a metologia de pesquisa utilizada para a revisão e a conclusão. A revisão foi feita para conhecer trabalhos relacionados aos conceitos dessa dissertação.

4.1 Conceito

Para o desenvolvimento deste trabalho foi feita uma revisão sistemática, que é, segundo Kitchenham et al. (2009), uma revisão metodologicamente rigorosa que destina-se a apoiar o desenvolvimento de diretrizes baseadas em evidências para os profissionais. Para isso, utiliza um protocolo de revisão que é uma especificação dos métodos, que serão usados no decorrer da revisão.

Seguindo as diretrizes de Kitchenham et al. (2009), a metologia se baseia em três tópicos, que foram desmembrados para melhor entendimento:

- **Questões de pesquisas:** consiste no planejamento da revisão. São especificadas questões que darão uma direção ao processo de busca e é descrito um protocolo de revisão;
- **Estudos primários:** consiste na busca por palavras-chave em bases científicas selecionadas e análise dos resultados. Para esse trabalho foram selecionadas as bases científicas IEEE Explore e ACM Digital Library, por serem disponibilizadas para alunos da Unesp. Para seleção de estudos primários entre os candidatos foram consideradas a relevância das palavras-chave e adequação às questões de pesquisa; e
- **Conclusão:** consiste na sintetização dos dados levantados.

Os autores propõem protocolos de revisão, os quais representam passos a serem seguidos para o andamento da revisão. Nesse trabalho foi utilizado o protocolo PR1, que abrange: questões de pesquisa, busca por estudos primários em bases científicas, extração de dados candidatos e seleção dentre os estudos primários, segundo o autor utilizado como referência para a revisão.

Para este trabalho, foram feitas duas pesquisas, uma sobre Bibliotecas e Dependências e outra sobre Visualização da Informação, porque não está apresentada uma pesquisa unindo as duas áreas porque não foi encontrado um número de trabalhos relevantes para seguir o exemplo dos autores Kitchenham et al. (2009).

4.2 Bibliotecas e Dependências

Esta seção apresenta uma revisão sistemática abrangendo Bibliotecas e Dependências.

4.2.1 Questões de pesquisa

Tendo em vista que é necessário conhecer os trabalhos desenvolvidos até aqui que tiveram como objetivo analisar bibliotecas e dependências de um projeto, foram definidas as questões de pesquisa:

1. **Foram desenvolvidas aplicações para mapeamento de bibliotecas e dependências?**
2. **Como é feita a análise do mapeamento realizado nas bibliotecas e dependências?**

4.2.2 Estudos primários

Conforme o autor escolhido como referência para a revisão sistemática, (KITCHENHAM et al., 2009), pode ser criada uma lógica de busca por meio de termos selecionados e operadores lógicos (AND e OR, nesse caso). Para a busca referente a bibliotecas e dependências de software foi utilizada a lógica de busca representada como LB1:

LB1: (("third-party libraries"AND "dependencies"AND ("software development"OR "software reuse")))

Baseado no protocolo escolhido, PR1, a Tabela 2 apresenta a documentação da busca nas bases científicas. Para análise de dados atuais, foram selecionados trabalhos a partir do ano de 2010 até o ano atual.

Tabela 2 – Documentação da Busca nas Bases Científicas Definidas para LB1.

Base Científica	**Lógica de Busca**	**Intervalo de Ano**	**Data**
IEEE Xplore	LB1	2010-2014	03/11/2014
ACM Digital Library	LB1	2010-2014	03/11/2014

O resultado da busca está descrito na Tabela 3.

Tabela 3 – Resultados da Busca para LB1.

Base Científica	**Produção Científica**
IEEE Xplore	35
ACM Digital Library	33
Total	68
Candidatos	24
Estudos Primários	12

4.2.3 Considerações Finais

Na busca em bases científicas foram selecionados 24 candidatos a estudos primários, os quais foram listados no Apêndice.

Todos os trabalhos selecionados usaram a linguagem de programação Java. Segundo Dietrich, Jezek e Brada (2014), a ampla utilização do Java deve-se às suas características como pacotes, classes herdadas, arquivos no formato JAR e disponibilidade de interfaces.

Também é notável que os trabalhos em geral utilizam tabelas e grafos para apresentar o conjunto de dados que é objeto de estudo, porém alguns utilizam outras formas de visualização, que são destacadas abaixo e interessantes para este trabalho.

Em todos os trabalhos selecionados, os autores citam a utilização em larga escala e a importância do uso de bibliotecas de terceiros em sistemas.

Para responder às questões de pesquisa, segue análise delas, respectivamente:

- **Questão 1:** 12 trabalhos, que correspondem à 50%, fazem mapeamento de bibliotecas e dependências.
- **Questão 2:** 7 trabalhos, que correspondem à 30%, apresentam dados com gráficos, os outros apenas em tabelas e grafos.

4.3 Visualização da Informação

Esta seção apresenta uma revisão sistemática abrangendo Visualização da Informação com ênfase em Sistemas de Informação e Engenharia de Software.

4.3.1 Questões de pesquisa

Tendo em vista que é necessário conhecer os trabalhos desenvolvidos até aqui que tiveram como objetivo analisar a engenharia, manutenção ou desenvolvimento de um projeto, foram definidas as questões de pesquisa:

1. **Apresenta uma representação para visualização dos dados?**
2. **Dos trabalhos que possuem uma representação, quais tem como foco principal bibliotecas e dependências?**

4.3.2 Estudos primários

Para análise de trabalhos relacionados a Engenharia de Software ou Desenvolvimento/Manutenção de Software e Visualização da Informação, pode ser criada uma lógica de busca por meio de

termos selecionados e operadores lógicos (AND e OR, nesse caso). Foi utilizada para essa pesquisa uma lógica de busca representada como LB2:

LB2: (("software development"OR "software reuse"OR "software engineer") AND ("visualization information"OR "visualisation information"))

Baseado no protocolo escolhido, PR1, a Tabela 4 apresenta a documentação da busca nas bases científicas. Para análise de dados atuais, foram selecionados trabalhos a partir do ano de 2010 até o ano atual. Porém foram encontrados artigos apenas até 2013.

Tabela 4 – Documentação da Busca nas Bases Científicas Definidas para LB2.

Base Científica	**Lógica de Busca**	**Intervalo de Ano**	**Data**
IEEE Xplore	LB2	2010-2013	13/05/2015
ACM Digital Library	LB2	2010-2013	20/05/2015

O resultado da busca está descrito na Tabela 5.

Tabela 5 – Resultados da Busca para LB2.

Base Científica	**Produção Científica**
IEEE Xplore	31
ACM Digital Library	12
Total	43
Candidatos	26
Estudos Primários	17

4.3.3 Considerações Finais

Na busca em bases científicas foram selecionados 26 candidatos a estudos primários por abordarem como tema VI e/ou bibliotecas e dependências. Estes trabalhos foram listados no Apêndice.

Nos trabalhos candidatos destaca-se a necessidade aumentar o entendimento sobre a construção do sistema no aspecto de desenvolvimento ou mesmo de seu funcionamento. Dentre os 26, 22 buscavam esse aumento por meio da VI.

Visando analisar quantitativamente à revisão, utilizaremos às questões de pesquisa:

- **Questão 1:** 17 trabalhos, que correspondem a mais que 60%, apresentam uma representação para visualização dos dados.
- **Questão 2:** Apenas 2 trabalhos, entre os 17 que possuem uma representação, que correspondem a, aproxidamente, 12%, tem como foco as bibliotecas e dependências de um projeto.

4.4 Trabalhos relacionados

Com as pesquisas realizadas destacaram-se dois trabalhos.

O primeiro foi um artigo de Raemaekers, Deursen e Visser (2012) e destacou-se por apresentar métricas para analisar uma biblioteca, e pôde ser utilizado nesta dissertação para agregar valor à representação.

O segundo foi um plugin para o Eclipse, que pode ser utilizado para o denvolvimento de softwares em Java e destacou-se por apresentar as dependências do software que é o intuito do trabalho atual.

4.4.1 Measuring Software Library Stability Through Historical Version Analysis

O objetivo do trabalho de Raemaekers, Deursen e Visser (2012) é avaliar quão estável é um sistema de informação que usa um conjunto de bibliotecas de terceiros. Para isso, foram extraídas as dependências do arquivo de configuração do Maven e foram analisados o código das bibliotecas e o sistema.

Ao decorrer de toda essa seção serão detalhados pontos de vista dos autores e suas conclusões, portanto toda ela faz referência a eles.

Novas versões das bibliotecas utilizadas em um sistema podem ser lançadas a qualquer momento, quando uma nova versão corrige problemas que estavam ocorrendo na predecessora os desenvolvedores são obrigados a atualizar a biblioteca e os impactos dessa atualização do projeto podem ser muitos, como apresentado na Figura 15 (RAEMAEKERS; DEURSEN; VISSER, 2012).

A motivação dos autores foi o fato de que se o número de parâmetros dos métodos da biblioteca mudar, os desenvolvedores que usam esta biblioteca não terão escolha, terão que alterar todas as chamadas desses métodos em seu sistema. Raemaekers, Deursen e Visser (2012) propuseram um caminho para analisar a interface e a estabilidade da implementação analisando valores históricos de métricas medidos por chamadas de métodos, classes ou pacotes que estão sendo usados.

Como exemplo da necessidade dessa análise, foi apresentado um caso real de uma aplicação que depende das bibliotecas Spring, Apache Struts e Hibernate, tendo em vista que esses são frameworks que auxiliam o desenvolvimento de software, qualquer modificação feita em qualquer uma delas pode ter um impacto muito grande em um sistema. Neste caso o sistema foi desenvolvido em 2004 e utilizou a versão mais atual das bibliotecas até então. Analisando o sistema e suas dependências no ano de 2008, grandes alterações foram feitas no Spring, até mesmo seu nome foi alterado para Spring Security Framework. Essas alterações e correções se não fossem atualizadas no sistema poderiam comprometer a segurança do projeto. Com a

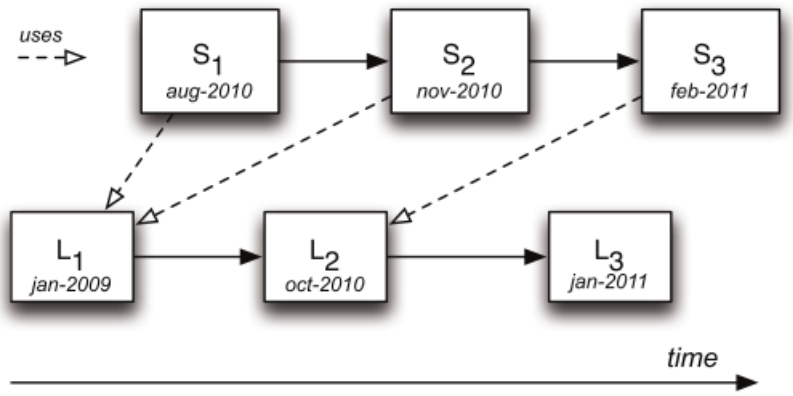

Figura 15 – Exemplo de relacionamento entre diferentes versões de um sistema e diferentes versões de bibliotecas que ele usa. S1 representa a primeira versão do sistema feita em Agosto de 2010. Esta versão usa a biblioteca L1 com data de Janeiro de 2009. A próxima versão do sistema, S2, continua usando a versão L1 da biblioteca, enquanto já foi lançada uma nova versão daquela biblioteca lançada em Outubro de 2010.

Fonte: Raemaekers, Deursen e Visser (2012).

atualização do framework do Spring no sistema, foi verificado que essa versão não é compatível com a versão que está sendo utilizada do Struts e que, portanto, esta também terá que ser atualizada. Dentre as mudanças está o uso da linguagem de expressão Java Server Pages (JSP) e todas as páginas web terão que ser alteradas para a nova sintaxe (RAEMAEKERS; DEURSEN; VISSER, 2012).

Para este trabalho, foi considerado que uma API (Application Programming Interface) ou biblioteca é estável se não teve funcionalidades removidas de sua interface. Para demonstrações foi usado um sistema desenvolvido em Java, utilizando a ferramenta Maven, já citada anteriormente.

Segundo os autores, ferramentas já haviam sido propostas para detectar a evolução de APIs e sugerir as modificações que seriam necessárias no caso de uma substituição de biblioteca, mas o diferencial desse trabalho é que os autores buscam usar esse tipo de informação para auxiliar na tomada de decisão de incluir ou não determinada biblioteca no sistema, ou mesmo escolher uma melhor alternativa entre as disponíveis.

Como metodologia foram selecionadas métricas que tiveram seus valores analisados por algumas versões (no mínimo três versões, para poder analisar, no mínimo, dois valores, já que cada valor é extraído da comparação entre duas versões). São estas as métricas:

- WRM: basea-se no número de métodos excluídos em S_1, com relação a S_0. Sendo S, "snapshot", ou seja, versão da biblioteca.
- CEM: refere-se as alterações realizadas nos métodos comuns entre as versões analisadas.
- RCNO: relação entre as métodos novos e existentes entre as versões analisadas.

- PNM: calcula a porcentagem de novos métodos entre as versões analisadas.

O cálculo das métricas é feito da forma descrita na Figura 16:

$$U_{o(s,s+1)} = U_s \cap U_{s+1} \quad (1)$$
$$U_{n(s,s+1)} = U_{s+1} \setminus U_s \quad (2)$$
$$U_{r(s,s+1)} = U_s \setminus U_{s+1} \quad (3)$$

$$hw(s) = \frac{1}{2^{s-1}} \quad (4)$$

$$\Delta U(s,s+1) = \frac{1}{\sum_{u \in \Delta U} w_u} \sum_{u \in \Delta U} w_u |M_{u,s+1} - M_{u,s}| \quad (5)$$

$$R = \sum_{s=1}^{|S|-1} hw_s R_x \quad (6)$$

$$WRM = \sum_{u \in U} w_u U_r \qquad CEM = \Delta U_o \quad (7,\ 8)$$

$$RCNO = \frac{\Delta U_n}{\Delta U_o} \qquad PNM = \frac{|U_n|}{|U_o| + |U_n|} \quad (9,\ 10)$$

Figura 16 – Fórmulas para cálculo das métricas.

Fonte: Raemaekers, Deursen e Visser (2012).

Na Tabela 6, estão descritas as variáveis utilizadas.

Tabela 6 – Descrição dos símbolos das equações

Símbolo	**Explicação**
u	Uma unidade em específico, sendo unidade: pacote, classe ou método.
U	Conjunto de todas unidades de um sistema.
s	Uma versão da biblioteca.
S	Conjunto de versões de uma biblioteca.
U_s	Todas unidades de determinada versão.
$U_o(old)$	Todas unidades de $s+1$ que estão em s.
$U_n(new)$	Todas unidades de $s+1$ que não estão em s.
$U_r(new)$	Todas unidades de s que foram removidas em $s+1$.
$hw(s)$	Variação dos valores historicamente de cada snapshot.
$M_(s,u)$	O valor da métrica M da unidade u no snapshot s.
w_u	Número de vezes que a unidade é referenciada no projeto.

Analisando o trabalho de Raemaekers, Deursen e Visser (2012) e relacionando-o ao trabalho atual, é importante destacar:

- Extração de dependências do software a partir do arquivo de configuração do Maven;
- Atualização constante de bibliotecas e dependências do software; e
- Análise do código das bibliotecas de software.

4.4.2 M2Eclipse

O M2Eclipse é um exemplo de software que gerencia as dependências de projetos desenvolvidos em Java. É um plugin para o Eclipse que, como apresentado em M2Eclipse (2014), detalha as dependências de um projeto de software:

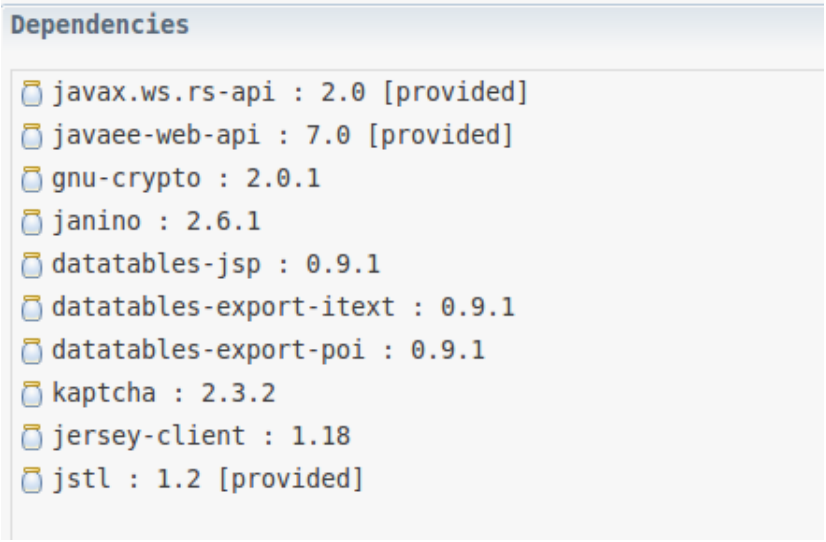

Figura 17 – Visualização das dependências de um projeto.

Fonte: Da autora.

O plugin possibilita ao desenvolvedor a visualização das dependências diretas e indiretas do projeto (M2ECLIPSE, 2014). Ou seja, além de apresentar as bibliotecas que o projeto utiliza, o plugin apresenta também as dependências delas, e assim, sucessivamente, como apresentado na Figura 18, em que pode-se notar a biblioteca *commons-compiler* que é uma dependência da biblioteca *janino* que é dependência do projeto selecionado.

Analisando o plugin e relacionando-o ao trabalho atual, é importante destacar:

- Visualização de todas as dependências de um projeto em uma só tela; e
- A importância de conhecer a versão do biblioteca utilizada, que é apresentada na sequência do nome, como pode-se notar na Figura 17.

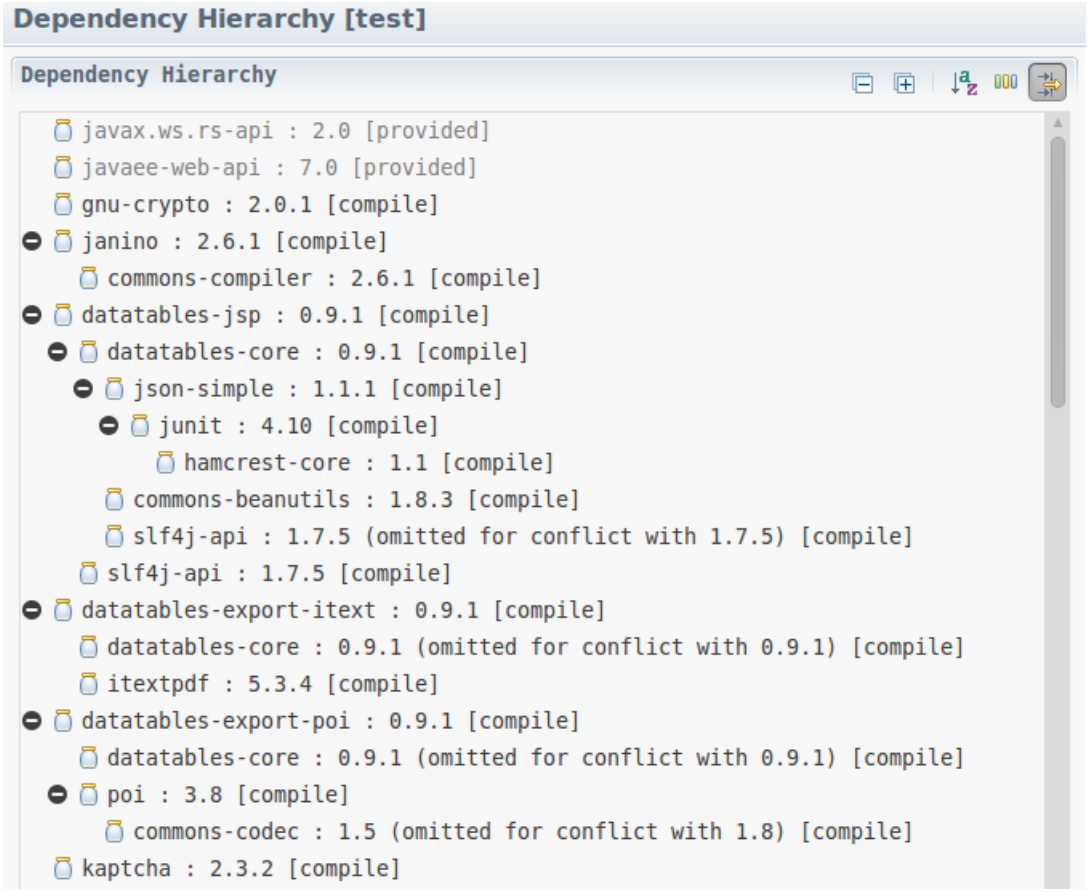

Figura 18 – Demonstração hierárquica das dependências explícitas e implícitas de um projeto.

Fonte: Da autora.

4.5 Considerações finais

Na pesquisa realizada, que tem seus principais resultados apresentados neste documento, foram identificados trabalhos atuais que possuem como objetivo principal o entendimento de bibliotecas. É possível concluir que esses trabalhos tem grande utilidade devido a ampla utilização das bibliotecas e o avanço contínuo das mesmas.

Algumas ferramentas apresentadas nos trabalhos analisados possibilitam o mapeamento e gerenciamento de bibliotecas, mas é possível explorar ainda mais a VI através de representações gráficas, disposição dos elementos e cores. Alguns autores optaram apenas por mostrar os dados em tabela, outros com grafos, poucos com gráficos.

Tendo em vista essas conclusões, diretrizes puderam ser estipuladas para o desenvolvimento da ferramenta dessa dissertação:

- Atualmente, as bibliotecas são amplamente utilizadas;
- Há a necessidade de conseguir analisar as dependências das bibliotecas em um sistema;
- As bibliotecas são, constantemente, atualizadas e novas versões são disponibilizadas;

- Métricas podem ajudar a analisar uma biblioteca; e
- Técnicas de VI podem ser utilizadas de forma simples e intuitiva.

5 LibViews

Com base nas pesquisas realizadas, o LibViews foi desenvolvido e será apresentado nesta seção.

O LibViews disponibiliza uma visualização simples e intuitiva de um projeto de software e suas dependências, que permite o entendimento rápido dos vínculos entre pacotes do software e bibliotecas, possibilitando, assim, melhor gerenciamento das dependências que o projeto possui.

O aplicativo apresenta também detalhes de cada biblioteca utilizada no projeto, auxiliando o desenvolvedor na tomada de decisões sobre qual biblioteca e qual versão dela utilizar no software em desenvolvimento ou em manutenção.

A Figura 19 mostra as etapas do funcionamento do LibViews.

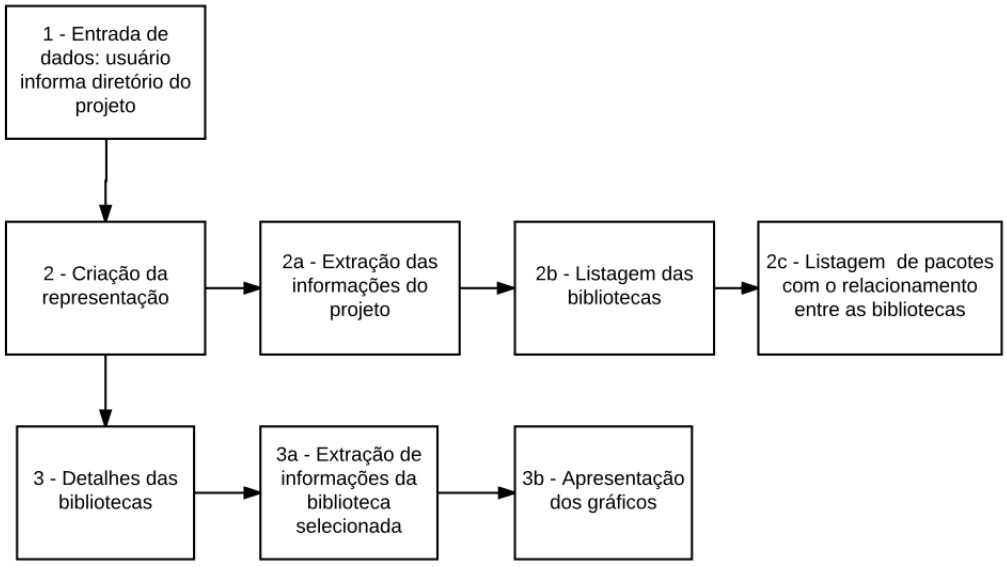

Figura 19 – Fluxograma descrevendo o funcionamento do LibViews.

Fonte: Da autora.

5.1 Descrição Técnica

A seguir, estão apresentados detalhes das tecnologias utilizadas no desenvolvimento de software.

Tendo em vista que todos os trabalhos conhecidos através da pesquisa utilizam o Java como linguagem de programação, essa também foi a escolha para a aplicação. E, a ferramenta Maven disponível para a linguagem Java, pôde ser utilizada para o gerenciamento de bibliotecas dos projetos.

Com intuito de disponibilizar o LibViews para o maior número de usuários possíveis e de qualquer lugar, web foi a plataforma escolhida. Ou seja, a aplicação pode ser acessado de qualquer lugar a partir de computadores ou dispositivos móveis.

E para a criação da representação, foi utilizada a biblioteca D3.js.

5.1.1 Maven

Maven é uma ferramenta de compilação popular para desenvolvimento de projetos em Java. Dentre suas principais características estão: usabilidade, multiplataforma, conhecimento das dependências do projeto para baixá-las e compilação automática (ZHEN-HAI; YONG-ZHI, 2014). Sendo que, ferramentas de compilação são responsáveis por transformar códigos estáticos em sistemas executáveis (MCINTOSH; ADAMS; HASSAN, 2010). A Figura 20 apresenta as etapas de compilação do Maven.

O movimento de código aberto tem mudado fundamentalmente o processo de desenvolvimento de software para muitos praticantes. A quantidade de código fonte gratuito disponível na web tem tornado viável alternativas a construção de programas inteiramente do zero. Há uma variedade de desafios que esta abordagem apresenta aos desenvolvedores, que vão desde a localização de artefatos relevantes para compreendê-los e integrá-los em um novo sistema. Com o intuito de auxiliar nesses desafios o Maven foi criado (OSSHER; BAJRACHARYA; LOPES, 2010).

Como descrito em Maven (2014), os objetivos do Maven são:

- Simplificar a compilação do projeto;
- Criar um padrão de compilação de projeto;
- Garantir a qualidade das informações do projeto;
- Dar diretrizes para melhores práticas no desenvolvimento; e
- Permitir uma migração transparente para novas funcionalidades.

Essa ferramenta gerencia a compilação, acompanhamento e documentação dos projetos baseado num Project Object Model (POM). POM é o núcleo do Maven, é um arquivo XML que contém as informações sobre as configurações, construção do ambiente e dependências do projeto. Na Figura 21 está apresentado um arquivo POM básico.

Este gerenciador de biblioteca possui um repositório central remoto onde são disponibilizadas as bibliotecas de uso público. Qualquer usuário pode fazer o upload de dependências desde que siga os requisitos estabelecidos, dentre eles estão: *JavaDoc*, *groudId*, *artifactId*, *version*, nome, descrição e url do projeto. Deste repositório remoto são copiadas para um repositório

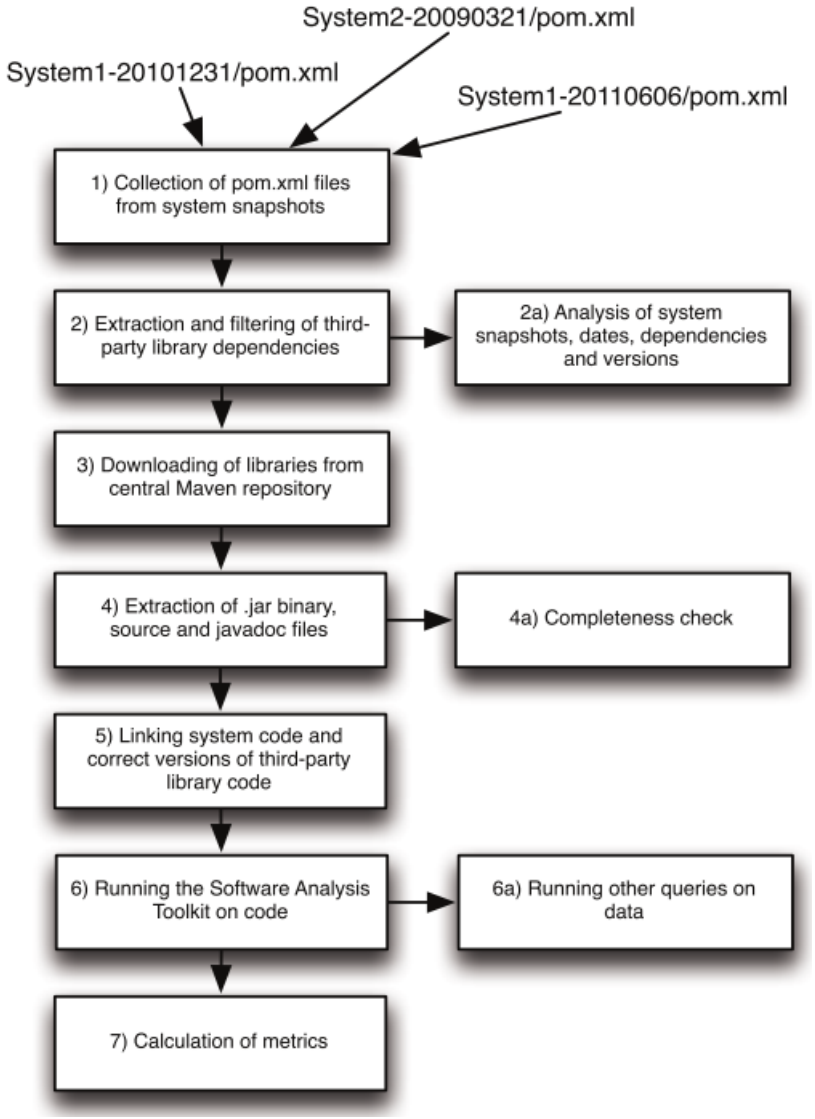

Figura 20 – Processo de mineração de um repositório Maven.

Fonte: Raemaekers, Deursen e Visser (2012).

local do desenvolvedor as bibliotecas utilizadas. Como apresentado na Figura 21, os campos groupId, artifactId e version formam a chave para identificar a biblioteca no repositório público do Maven. Após a identificação das bibliotecas pode ser feito o download para o servidor local do projeto como apresentado na Figura 22 (MAVEN, 2014).

Segundo Raemaekers, Deursen e Visser (2012), cada projeto Maven contém um arquivo "pom.xml"que pode conter uma seção para listagem de dependências com informações como nome e versão das bibliotecas utilizadas (Figura 23).

5.1.2 Frameworks e linguagens Web

Para o desenvolvimento do LibViews foram seguidos os padrões da web para acessibilidade definidos pela W3C com o objetivo de garantir a condição de utilização para todos os

```
<project>
        <modelVersion>4.0.0</modelVersion>
        <groupId>mi.autoUpdate.app</groupId>
        <artifactId>test-app</artifactId>
        <version>1.0</version>
</project>
```

Figura 21 – Arquivo POM básico.

Fonte: Zhen-hai e Yong-zhi (2014).

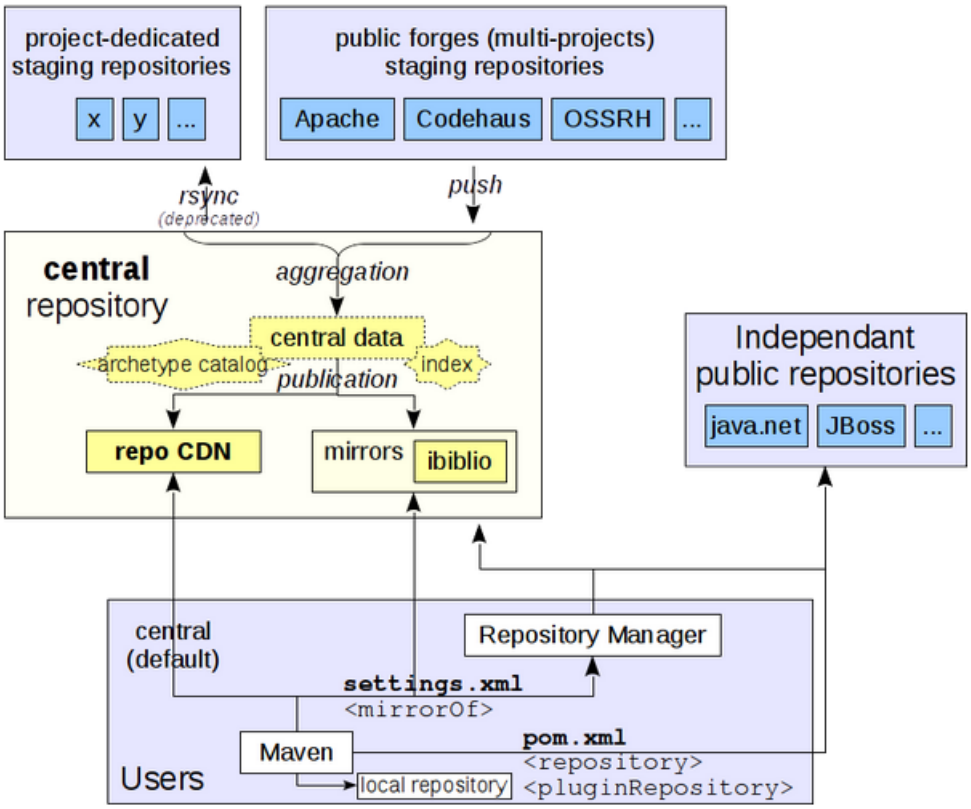

Figura 22 – Repositório do Maven (MAVEN, 2014).

usuários sem distinção. W3C é uma comunidade internacional que desenvolve padrões abertos para assegurar o crescimento a longo prazo da Web (W3C, 2015).

O design da aplicação é responsivo. Web design responsivo é um conjunto de técnicas aplicadas no layout de um projeto, que permite que um projeto web se adapte a qualquer dispositivo de acordo com a largura de tela (MAJID; KAMARUDDIN; MANSOR, 2015).

Além do Java, foram utilizadas a linguagem HTML, CSS e Javascript para a criação da aplicação.

Frameworks também foram utilizados para a criação da interface com o usuário (exemplos da utilização prática das bibliotecas), são eles: *Foundation*, *Primefaces*, *JSF* e *D3.js*.

```
<dependencies>
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.1</version>
  </dependency>
</dependencies>
```

Figura 23 – Exemplo da descrição de uma dependência no arquivo pom.xml.

Fonte: Raemaekers, Deursen e Visser (2012).

Foundation é um *framework* de front-end responsivo produzido pela ZURB (FOUNDATION, 2015). Ou seja, ele disponibiliza componentes formatados em CSS para a construção do layout da página. É chamado responsivo, pois adapta a visualização ao tamanho da tela do usuário, permitindo que a aplicação seja vista com a mesma qualidade quando acessada por um computador ou dispositivo móvel.

O *Primefaces* foi utilizado para gerar os gráficos e é uma biblioteca que oferece componentes de interface do usuário prontos para uso. É baseado em JSF. (PRIMEFACES, 2015b).

Java Server Faces (JSF) é uma tecnologia que simplifica a criação de interfaces para aplicações em Java (JSF, 2015). E, segundo Oracle (2015), inclui:

- Um conjunto de *APIs* para representar componentes de interface do usuário e gerenciamento de seu estado, manipulação de eventos e de validação de entrada, definindo navegação de página, e apoiar a internacionalização e acessibilidade; e
- Componentes personalizados que fazem o relacionamentos entre o código em Java e as páginas de interface em Java (JavaServer Pages - JSP).

5.1.3 D3.js

D3.js é uma biblioteca JavaScript para manipulação de documentos com base em dados que dinamiza dados usando HTML, CSS e SVG . A ênfase de D3.js em padrões web dá-lhe todas as capacidades de navegadores modernos sem amarrar-se com uma estrutura proprietária, combinando componentes de visualização poderosas e uma abordagem orientada a dados para manipulação DOM (D3, 2015). Sendo que:

CSS descreve como os elementos HTML serão mostrados na tela, no papel ou em outra mídia (W3SCHOOLS, 2016).

SVG é utilizado para criação de imagens vetorizadas para web usando XML, que não perdem qualquer qualidade se forem ampliada ou redimensionada, sendo que cada elemento e todos os atributos em arquivos SVG podem ser animados (W3SCHOOLS, 2016). Na Figura 24 é

apresentada uma página HTML (código à esquerda, e visualização à direita) que contêm uma imagem SVG e exemplifica a configuração de alguns atributos da imagem.

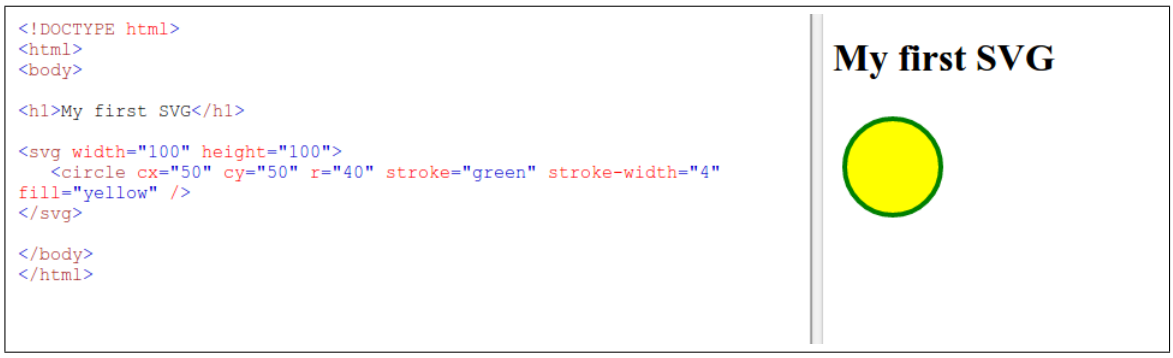

Figura 24 – Exemplo SVG.

Fonte: W3Schools (2016).

Definido pela W3C, DOM é uma interface que permite programas e scripts acessarem e atualizarem conteúdo, estrutura e estilo de um documento dinamicamente e é criado quando uma página web é carregada (W3C, 2015).

Há uma especificação do DOM para HTML, que a W3C (2015) define:

- Elementos HTML como objetos;
- Propriedades para todos elementos HTML;
- Métodos para acessar todos elementos HTML; e
- Eventos para todos elementos HTML.

O HTML DOM, como é chamado pode ser acessado usando JavaScript e na Figura 25 está descrita sua arquitetura.

Algumas vantagens da biblioteca D3.js (D3, 2015):

- Com sobrecarga mínima, é extremamente rápida;
- Pode manipular grandes conjuntos de dados e comportamentos dinâmicos de interação e animação;
- Padrão W3C e compatibilidade com navegadores e suas atualizações;
- Permite vincular dados arbitrários a um DOM e aplica neste documento transformações baseadas nos dados; e
- Podemos fazer novos templates ou editar já existentes.

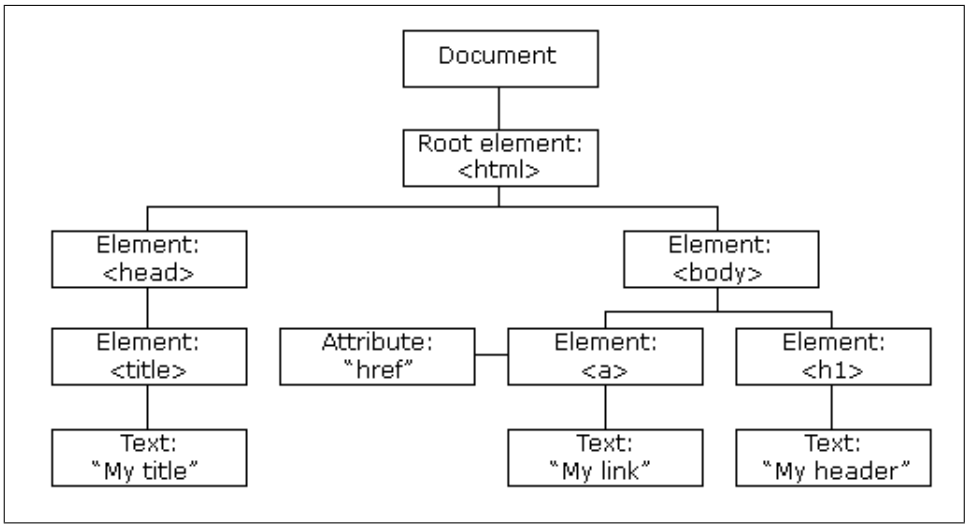

Figura 25 – Arquitetura do DOM.

Fonte: W3Schools (2016).

Na Figura 26, um gráfico é gerado a partir de dados importados pelo D3js [1]. E, na Figura 27, um Treemap é gerado a partir de dados importados pelo D3js [2]. Um treemap é uma representação capaz de organizar as informações de forma hierárquica de acordo com características delas, permitindo o acesso rápido às informações e destacando as informações mais relevantes (OLIVEIRA et al., 2014).

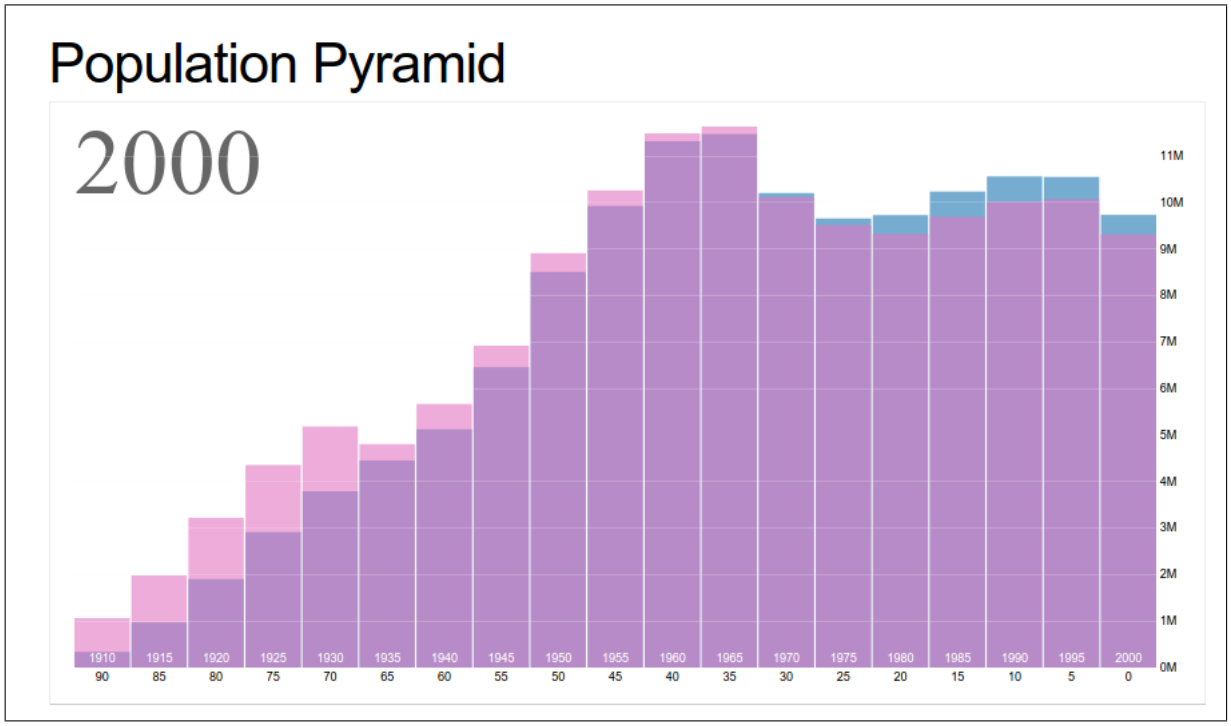

Figura 26 – Exemplo de gráfico usando D3.js.

Fonte: D3 (2016).

[1] Código fonte disponível em: http://bl.ocks.org/mbostock/4062085
[2] Código fonte disponível em: http://bl.ocks.org/mbostock/4063582

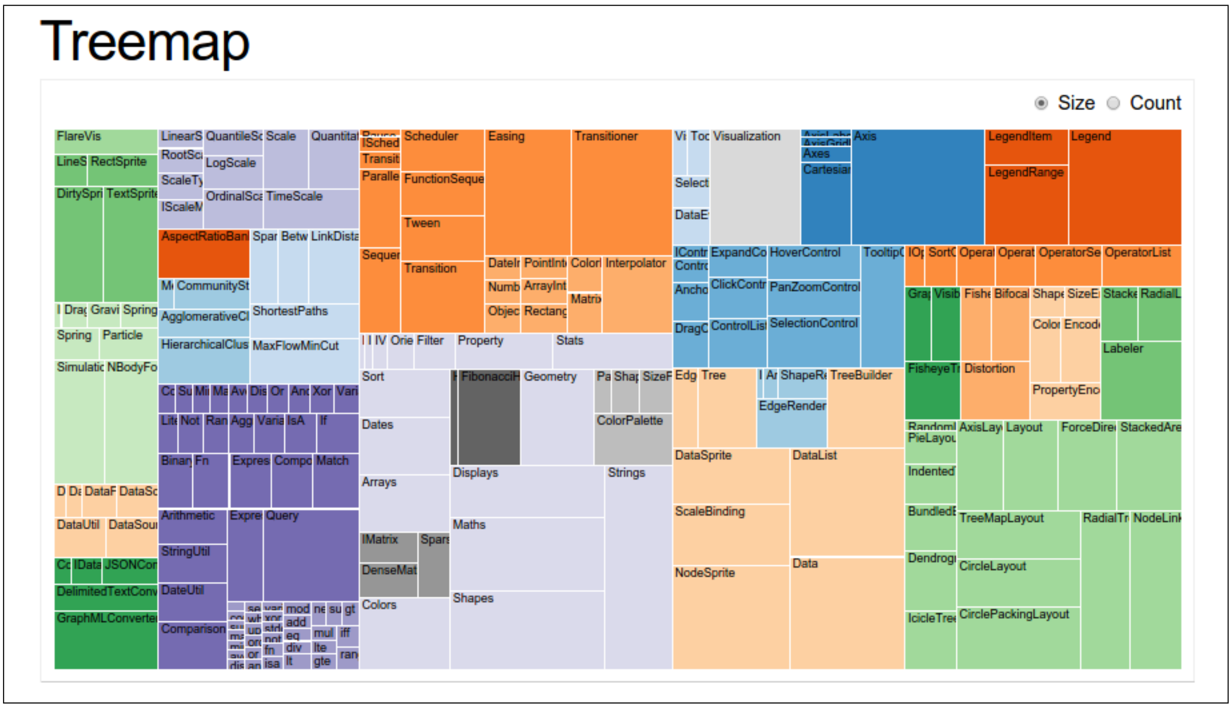

Figura 27 – Exemplo de Treemap usando D3.js.

Fonte: D3 (2016).

JSON é um formato para armazenar e transportar dados é usado para enviar dados do servidor para a web page. Sua sintaxe é derivada do JavaScript, mas é apenas texto (W3SCHOOLS, 2016). Na Figura 28 está descrito um exemplo de arquivo JSON com array de objetos com nome *employees*.

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

Figura 28 – Sintaxe do JSON.

Fonte: W3Schools (2016).

A biblioteca D3.js possui funções prontas para leitura de dados a partir de alguns tipos de arquivos, como CSV e JSON (formatos utilizados nos exemplos apresentados) (D3, 2016):

d3.csv(url, row, callback);

d3.request(url)

.mimeType("application/json")

.response(function(xhr) return JSON.parse(xhr.responseText);)

.get(callback);

5.2 Funcionalidades

Dividas em sub-seções, nesta seção estão descritas as funcionalidades principais do LibViews:

5.2.1 Formulário

A comunicação entre o usuário e o sistema ocorre por meio de um formulário, no qual pode ser selecionado o diretório do código fonte do projeto. A partir deste diretório, um algoritmo extrai as informações necessárias do projeto para gerar a representação, por isso deve ser descrito com seu caminho completo.

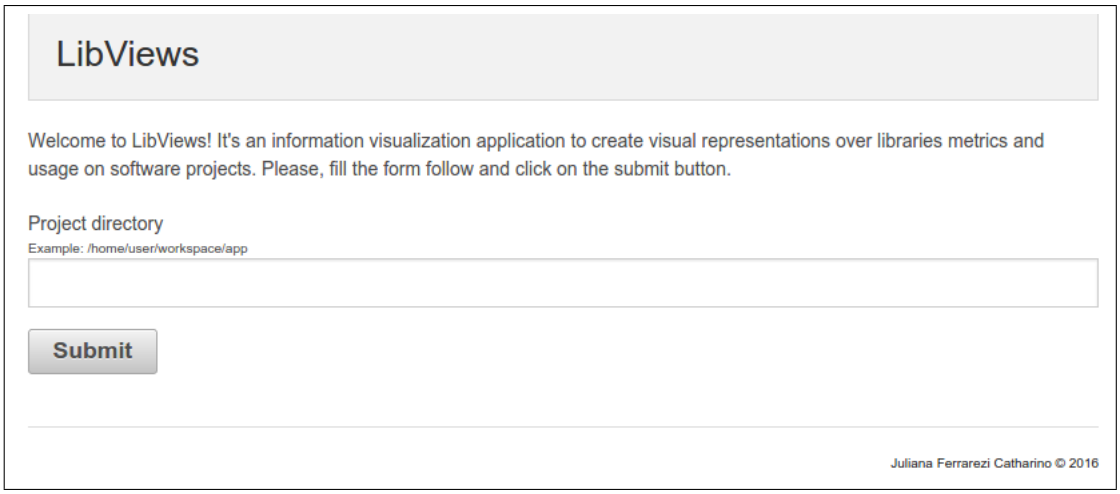

Figura 29 – Formulário do LibViews.

Fonte: Da autora.

Projetos escritos em Java que utilizam Maven poderam ser analisados com o LibViews e como em todo projeto que utiliza Maven, no diretório principal do projeto deve estar contido o arquivo de configuração *pom.xml*, do contrário, o LibViews informará que não é um projeto válido.

5.2.2 Extração das informações de um projeto de software

A partir de um algoritmo, são extraídas informações do projeto de software selecionado no formulário pelo usuários. Este algoritmo é dividido em duas etapas:

- Leitura do arquivo *pom.xml* para conhecer as bibliotecas que são as dependências do projeto; e
- Análise do código fonte do projeto para verificar quais bibliotecas são utilizadas em cada pacote do código Java.

Na primeira etapa é feita uma lista com todas as bibliotecas informadas no arquivo *pom.xml*. Já na segunda etapa, é feita uma varredura em todos os diretórios do código fonte, armazenando em estruturas de dados os pacotes e cada classe contida em cada um deles. Após essa análise, cada classe é analisada para verificar quais bibliotecas utilizam individualmente. Dessa forma, são armazenadas em estruturas de dados todas as dependências informadas para o projeto e quais bibliotecas são utilizadas em cada pacote Java.

5.2.3 Representação

Para gerar a representação das informações extraídas do projeto e suas dependências, foi utilizado um modelo disponível na galeria de exemplos da biblioteca D3js [3]. Chamado de *Concept network browser* o modelo foi escolhido por apresentar nomes ao centro e links para categorias ao redor, ou seja, mostra o relacionamento entre categorias diferentes, como mostra a Figura 30.

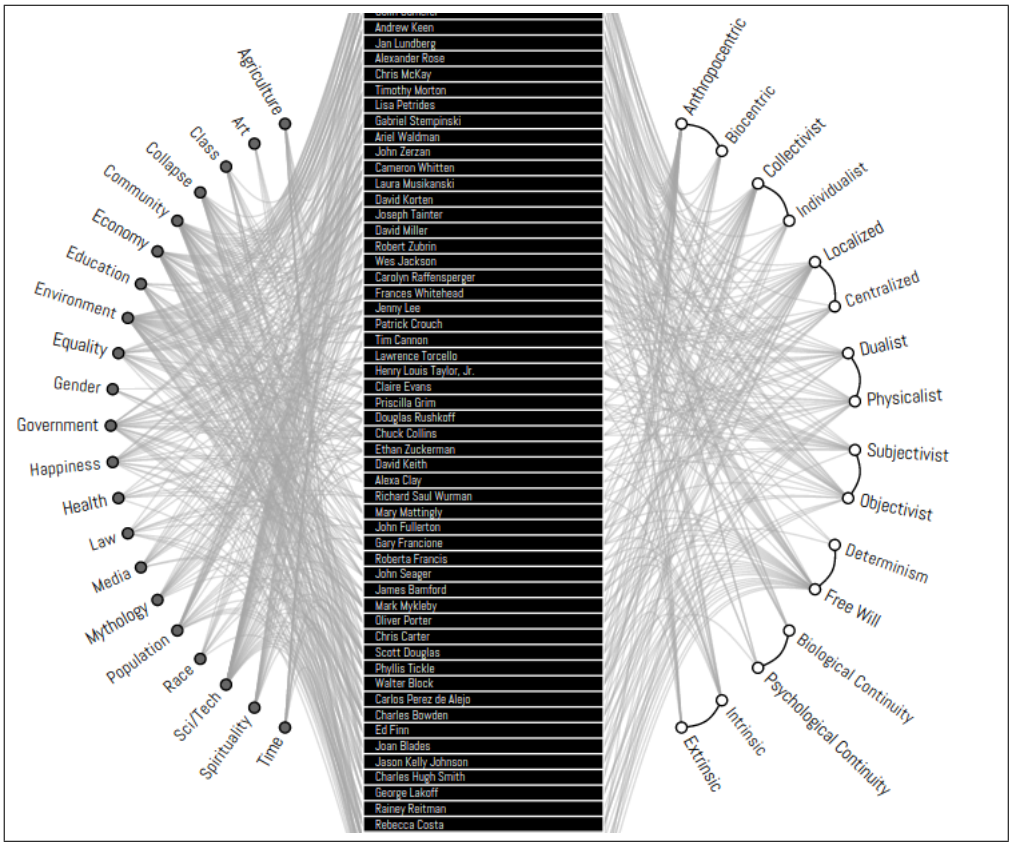

Figura 30 – *Concept network browser*.

Fonte: D3 (2016).

Na classificação descrita na Seção 3, é uma representação do tipo Rede Radial. Como apresentado na mesma seção, a representação utiliza-se de técnicas para melhorar o entendimento do usuário, como o uso de cor, legendas e animação.

[3] https://github.com/d3/d3/wiki/Gallery

Ao redor dos nomes apresentados no centro estão duas categorias diferentes: à esquerda, representada por círculos preenchidos uma; e, à direita, outra representada por círculos sem preenchimento. Ao clicar em quais dos elementos da tela, direciona para uma página de detalhes, como mostra a Figura 31.

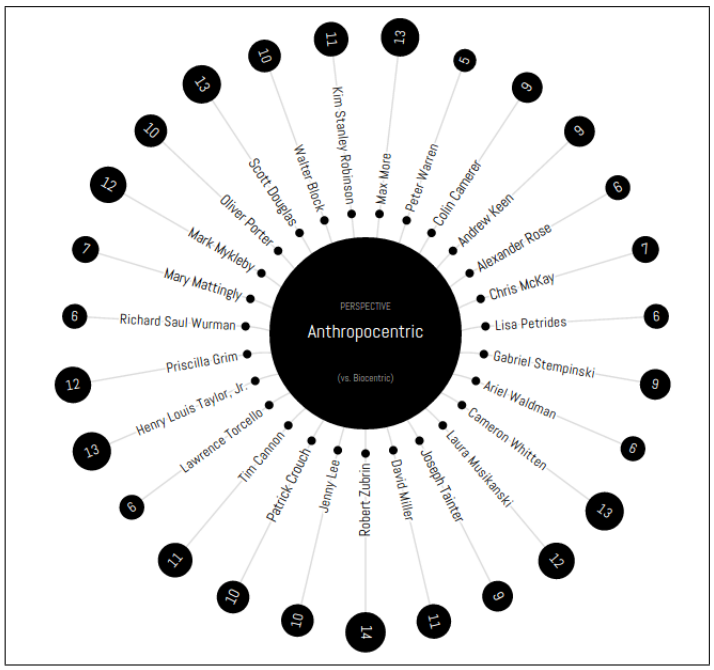

Figura 31 – *Concept network browser*: detalhes.

Fonte: D3 (2016).

Por essas peculiaridades, o modelo escolhido foi adaptado, para que apresentasse ao centro a listagem de pacotes do projeto analisado. E ao redor, todas as bibliotecas listadas no arquivo *pom.xml*. Os relacionamentos entre elas também são demonstrados e as dependências entre os pacotes e as bibliotecas são detalhadas colorindo as linhas que as ligam de azul, quando o mouse é posicionado sobre o pacote, ou sobre a biblioteca.

Também foi modificada a ação do clique nos componentes da tela: no LibViews, ao clicar nos pacotes do projeto nada é executado e quando o clique acontecesse nas bibliotecas são apresentados os gráficos com as métricas, detalhados na Seção 5.2.6.

Foi adicionada à representação a funcionalidade de colorir determinadas bibliotecas quando o método a biblioteca está listada como dependências mas não é utilizada no projeto, ela fica destacada com a cor laranja.

Como demonstração, foi gerada a representação de um sistema administrativo da Unesp, apresentada na Figura 32 [4]. Na Figura estão destacados com números:

1. Os pacotes do projeto;
2. As dependências do projeto;

[4] Essa representação está disponível no link: http://slstr.bauru.unesp.br/libviews/administrativo.xhtml

3. Os relacionamentos entre pacotes e bibliotecas;

4. As bibliotecas não utilizadas.

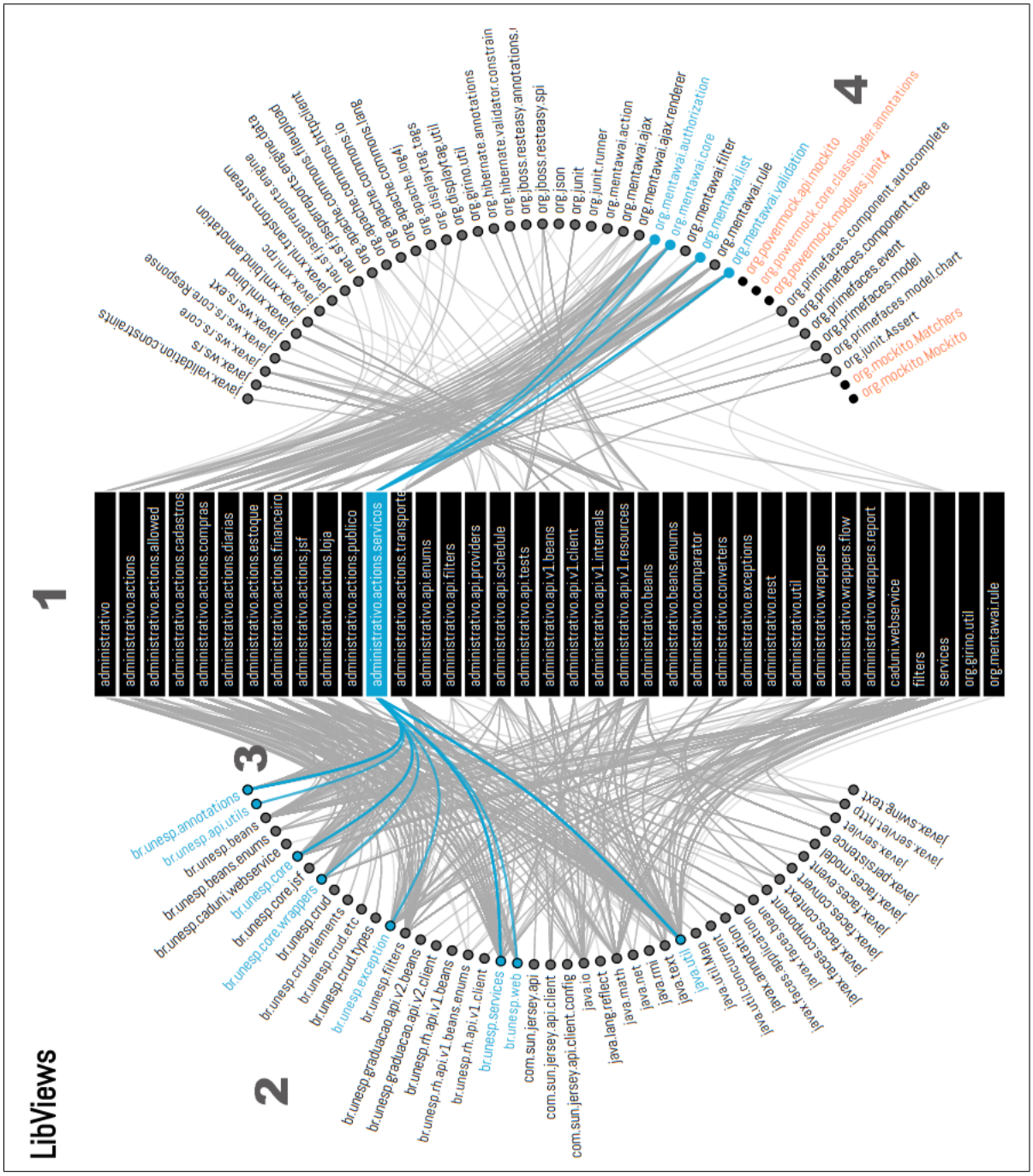

Figura 32 – Exemplo: Sistema Administrativo da Unesp.

Fonte: Da autora.

O uso do sistema da UNESP neste trabalho foi aprovado pelo Comitê Superior de Tecnologia da Informação (CSTI) da universidade, como consta na Súmula 73 [5].

[5] http://www.unesp.br/Home/csti/sumula-no73.pdf

5.2.4 Estudo das métricas

A fim de ter dados quantitativos já testados e utilizados, a aplicação se baseia nas métricas expostas na seção 4.4.1, a qual detalha o trabalho de Raemaekers, Deursen e Visser (2012) para avaliação de bibliotecas e dependências de software.

Os autores extraem informações das bibliotecas e dependências, para calcular quatro métricas (detalhadas na seção acima informada) que podem ser números de ocorrência de alguma varíavel candidata, ou mesmo porcentagens e relações, por exemplo: RCNO é descrito pela relação entre a variação de métodos novos e a variação de métodos antigos. Esse valor estará sempre no intervalo de 0 e 1.

Para a representação desenvolvida foram feitas modificações na utilização dos valores utilizados para os cálculos, pois são valores em escalas maiores que permitirão maior facilidade de identificação das variações na visualização.

As métricas serão utilizadas neste trabalho da seguinte forma, baseadas no trabalho dos autores:

- NRM: baseado no WRM, será o número de métodos removidos de cada versão. Permitirá ao usuário identificar se há constantes remoções de métodos, que é uma alteração que gera retrabalho nos projetos dependentes.
- NSM: baseado no CEM, será o número de métodos de cada versão, ao invés da relação entre esses valores, como os autores expõem. Dessa forma, gráficos possibilitarão a visualização da variação do número de métodos entre as versões.
- NNM: baseado no PNM, será o número de novos de cada versão, ao invés da porcentagem deles, como os autores expõem. Dessa forma, gráficos possibilitarão a visualização do número de métodos novos de cada versão, permitindo ao usuário da aplicação analisar a taxa de crescimento da API, conforme os autores propõem. Analisando esses valores, é possível ampliar a utilização da biblioteca no projeto e até mesmo, em alguns casos, eliminar a utilização de outra biblioteca que possui a mesma funcionalidade implantada nos métodos novos da biblioteca em questão.

Observação: a métrica RCNO, dos autores, não será utilizada, porque analisa o código de cada método internamente para calcular se a API está em estado em manutenção (alterações em métodos antigos) e/ou desenvolvimento (alterações em métodos novos) e o objetivo desse trabalho é analisar a biblioteca e sua dependência com um sistema de informação.

5.2.5 Extração de informações das bibliotecas

Tendo que o projeto selecionado utiliza o Maven e as bibliotecas utilizadas são copiadas para o repositório local pode ser feita uma análise de cada biblioteca.

Para essa análise foi utilizada a API *Reflection* que extrai informações do *JavaDoc* (documentação de sistemas desenvolvidos em Java) das classes e é usada para programas que precisam examinar, ou modificar, aplicações que estão sendo executadas na máquina virtual Java em tempo de execução.(API, 2015).

Dessa forma, estruturas de dados foram utilizadas para armazenar todas as classes e todos os métodos de cada classe de todas as bibliotecas listadas no *pom.xml* do projeto.

Essa análise se repetirá para todas as versões de cada biblioteca que já foram copiadas pelo Maven para o repositório local.

Foi desenvolvido um algoritmo que, baseado na listagem das classes e métodos de n versões de uma biblioteca, calcula os seguintes valores:

- Número de métodos removidos entre as versões ($M_r|S_0, S_1, ..., S_n$).
- Número de métodos de cada versão (M).
- Número de métodos adicionados entre as versões ($M_n|S_0, S_1, ..., S_n$).

A partir dos dados extraídos são calculadas as métricas da seguinte forma:

$$NSM = M_r|S_1, S_2, ..., S_n \tag{5.1}$$

$$NSM = M|S_0, S_1, ..., S_n \tag{5.2}$$

$$NNM = M_n|S_0, S_1, ..., S_n \tag{5.3}$$

Tabela 7 – Descrição dos símbolos das equações

Símbolo	**Explicação**
S	Conjunto de snapshots, ou seja, conjunto de versões das bibliotecas.
M	Conjunto de métodos de determinada versão da biblioteca.
M_r	Número de métodos removidos em S_n com relação a S_{n-1}.
M_n	Número de métodos novos em S_n com relação a S_{n-1}.

O algoritmo que calcula as métricas seguem os seguintes raciocínios:

- NRM: lê todas as classes e métodos da biblioteca e faz a análise de quais métodos foram removidos.
- NSM: lê todas as classes e métodos da biblioteca e informa o número de métodos.
- NNM: lê todas as classes e métodos da biblioteca e faz a análise de quais métodos foram adicionados.

Para verificar a confiabilidade do algoritmo que calcula as métricas, foram analisadas algumas classes selecionadas aleatoriamente da biblioteca Primefaces e comparadas com o JavaDoc da biblioteca disponibilizado em Primefaces (2015a).

A fim de realizar a verificação da métrica NRM, na Figura 33, está listado o resultado da análise dos métodos removidos da versão 5.1 da biblioteca, com relação a versão 5.0.

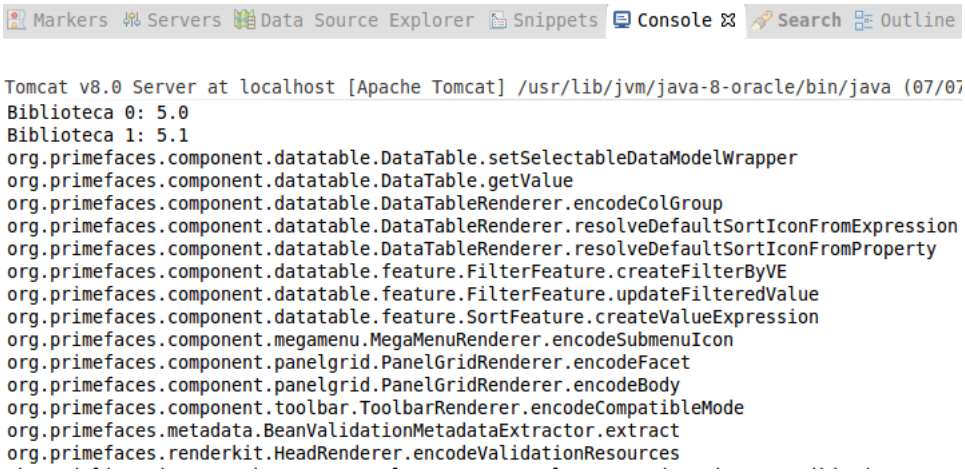

Figura 33 – Métodos removidos da biblioteca Primefaces versão 5.1 com relação a versão 5.0.

Fonte: Da autora.

Nas Figuras 34 e 35, é comprovado que o método “encodeColGroup“ existia na versão 5.0 e não existe na versão 5.1.

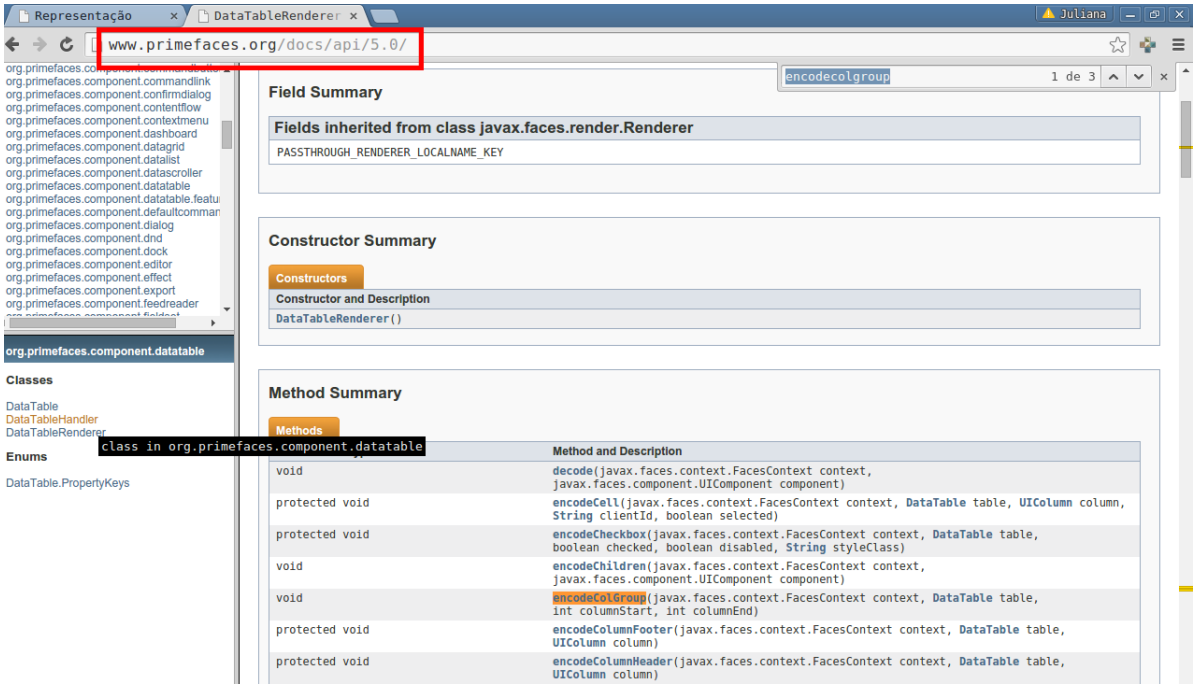

Figura 34 – Métodos da versão 5.0 da biblioteca Primefaces.

Fonte: Da autora.

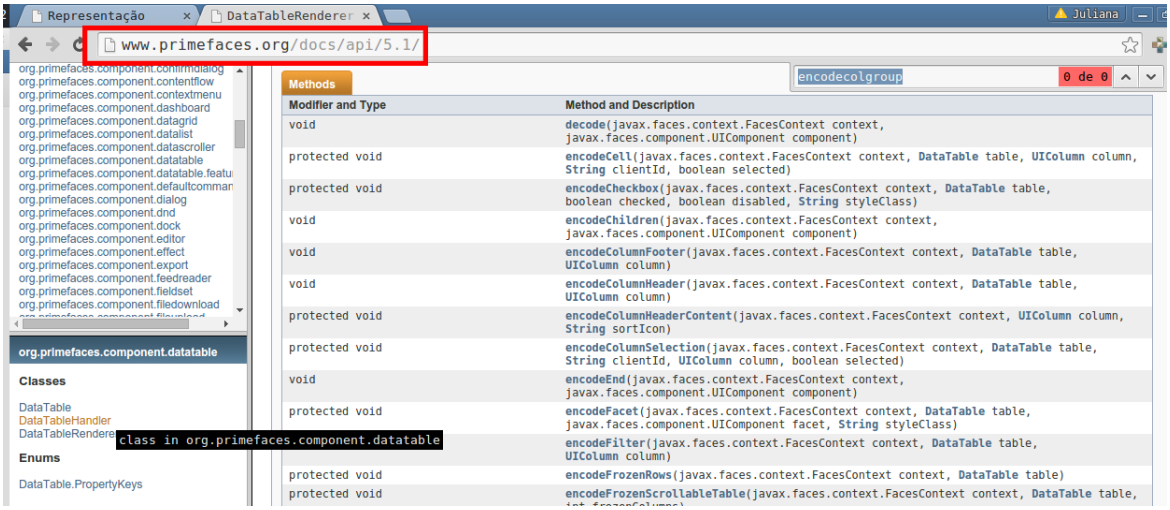

Figura 35 – Métodos da versão 5.1 da biblioteca Primefaces.

Fonte: Da autora.

A Figura 36 apresenta, à esquerda, classes que o pacote possui de acordo com o JavaDoc e, à direita, classes que o algoritmo encontrou, sendo que as destacadas são de um mesmo pacote.

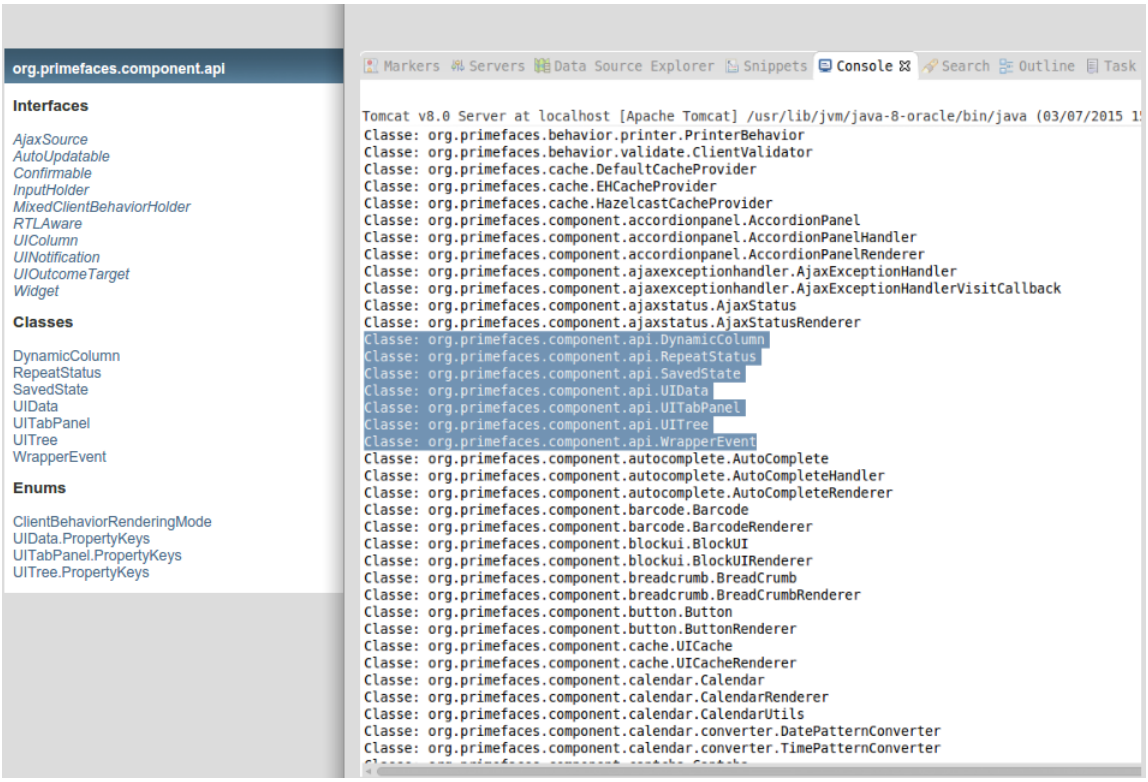

Figura 36 – Comparação das classes encontradas - JavaDoc e Algoritmo.

Fonte: Da autora.

A Figura 37 apresenta, à esquerda, métodos que a classe SavedState possui de acordo com o JavaDoc e, à direita, métodos que o algoritmo encontrou para essa classe.

Dessa forma, é garantida a extração correta das informações das bibliotecas para cálculo

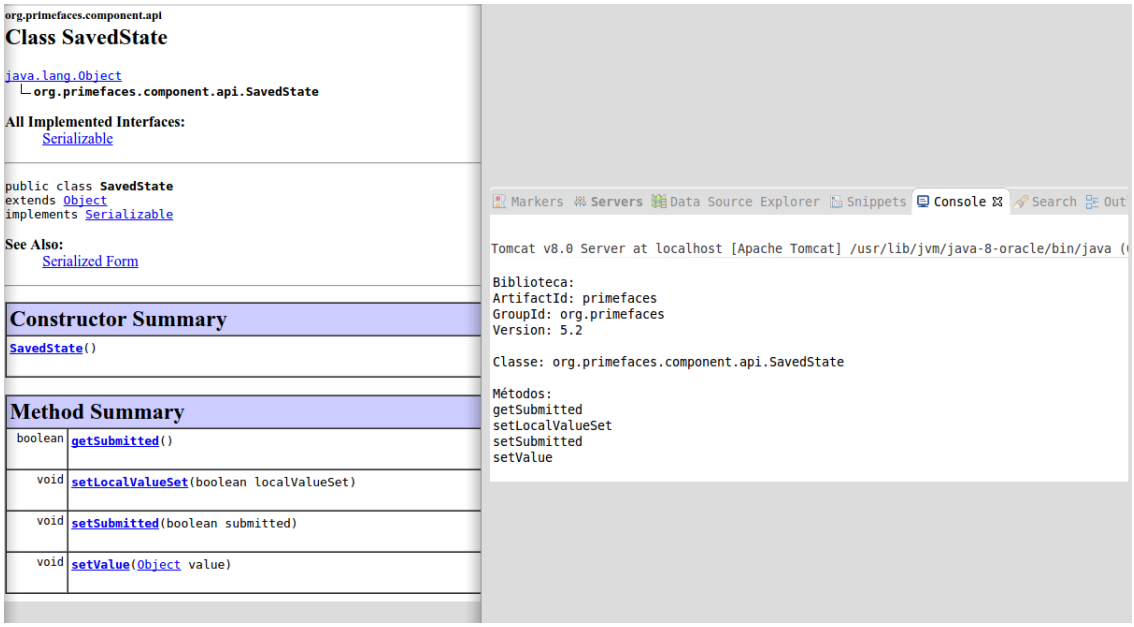

Figura 37 – Comparação os métodos encontrados - JavaDoc e Algoritmo.

Fonte: Da autora.

das métricas.

5.2.6 Gráficos

Na visualização detalhes de cada biblioteca, são apresentados gráficos baseados das métricas.

Para exemplificar, foram criados gráficos baseados em três versões da biblioteca Primefaces: 5.0, 5.1 e 5.2, representadas na aplicação, na sequência, por: 1, 2 e 3.

- NRM: apresentado na Figura 63, é possível afirmar que a biblioteca teve métodos removidos entre as versões 5.1 e 5.0, e também entre as versões 5.2 e 5.1.
- NSM: apresentado na Figura 39, é possível visualizar a variação do número de métodos de cada versão.
- NNM: apresentado na Figura 40, é possível afirmar que a biblioteca está em desenvolvimento contínuo, pois há métodos novos nas versões 5.1 e 5.2.

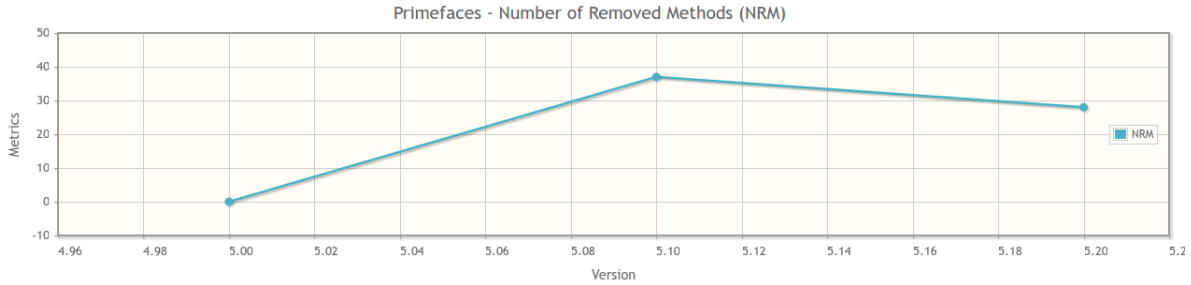

Figura 38 – Métrica NRM para as versões 5.0, 5.1 e 5.2 da biblioteca Primefaces.

Fonte: Da autora.

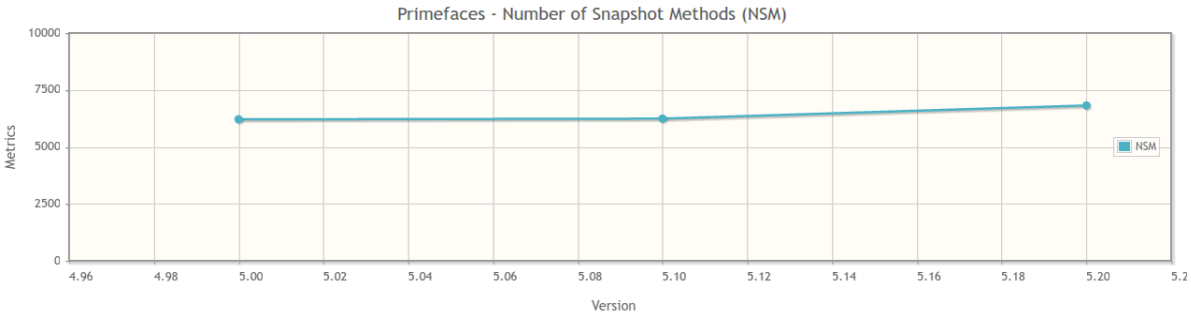

Figura 39 – Métrica NSM para as versões 5.0, 5.1 e 5.2 da biblioteca Primefaces.

Fonte: Da autora.

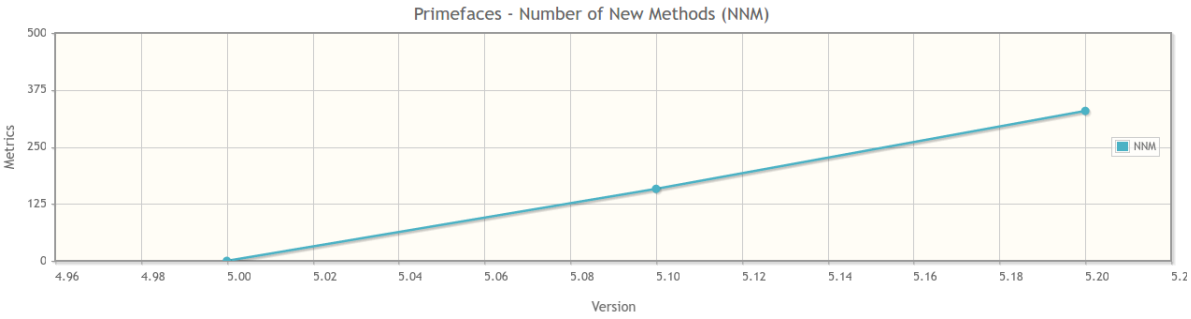

Figura 40 – Métrica NNM para as versões 5.0, 5.1 e 5.2 da biblioteca Primefaces.

Fonte: Da autora.

5.3 Considerações finais

Como visto no Capítulo 2, há necessidade de entender o relacionamento entre projetos de software e suas dependências. Também no Capítulo 3 concluiu-se que a VI pode melhorar a compreensão desses dados.

As pesquisas apresentadas no Capítulo 4 confirmaram e embasaram os conceitos e diretrizes para o desenvolvimento do LibViews que por meio de técnicas de VI mostra as informações de um projeto de software, suas dependências e detalhes das bibliotecas.

A aplicação conseguiu abordar todas as informações relevantes em uma só tela e os gráficos que apresentam as métricas detalham cada biblioteca e agregam valor à representação.

6 Resultados e Discussão

Este capítulo apresenta o questionário utilizado para avaliação do LibViews e as respostas dos usuários após o uso da aplicação, bem como as conclusões obtidas a partir da pesquisa.

6.1 Questionário

Foi disponibilizada para os usuários uma versão do LibViews com a representação do exemplo apresentado na Seção 5.2.3. Cada usuário pôde explorar os recursos da representação, como detalhamento dos relacionamentos entre os pacotes do projeto e suas dependências.

Para fazer a pesquisa com relação a utilização do LibViews pelos usuários, foi criado um questionário, descrito no Apêndice A.2, com perguntas elaboradas de acordo com as normas referentes a avaliação de software e diretrizes sobre usabilidade da Associação Brasileira de Normas Técnicas ABNT (2002) ABNT (2003).

A norma ISO 9241, que é usada na avaliação de usabilidade de sistemas interativos enfatizando o ponto de vista do usuário e seu contexto de uso, define:

1. Usuário como pessoa que interage com o produto;
2. Usabilidade como a capacidade de um produto ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso;
3. Eficiência como a precisão e completeza com que os usuários atingem seus objetivos, em relação à quantidade de recursos gastos; e
4. Satisfação - conforto e aceitabilidade do produto, medidos por meio de métodos subjetivos e/ou objetivos.

Tendo como possíveis respostas: excelente, muito bom, bom, ruim e muito ruim; ou sim e não; o questionário foi dividido em grupos de questões referentes às definições da norma e 21 usuários responderam.

6.1.1 Usuário

Questões foram aplicadas para conhecer o perfil do usuário e relacioná-lo com a experiência que ele teve de acordo com seu conhecimento.

Questão 1: **Conhecimento da linguagem de programação Java**.

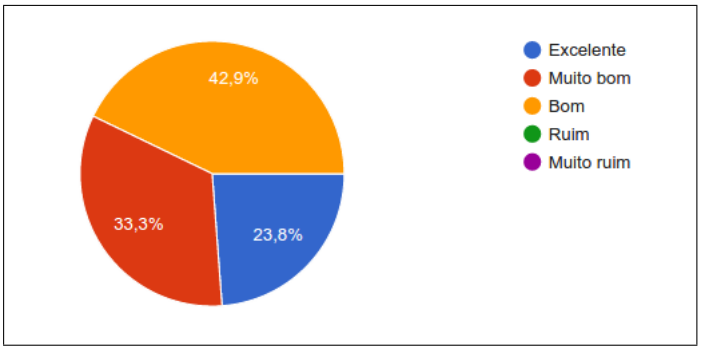

Figura 41 – Resposta da questão 1: Conhecimento da linguagem de programação Java.

Fonte: Da autora.

Com as respostas apresentadas na Figura 41, conclui-se que a maior parte dos usuários têm um conhecimento bom da linguagem de programação Java.

Questão 2: **Experiência com o uso de bibliotecas**.

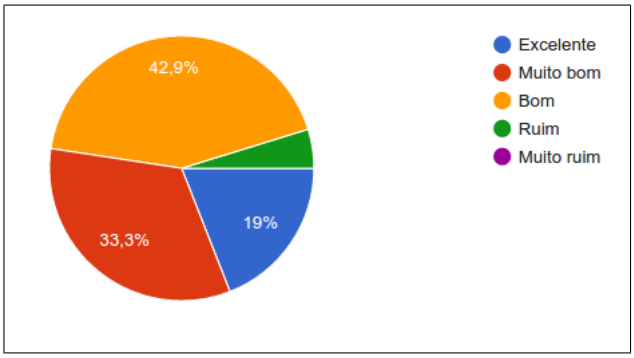

Figura 42 – Resposta da questão 2: Experiência com o uso de bibliotecas.

Fonte: Da autora.

Com as respostas apresentadas na Figura 42, conclui-se que todos os usuários já utilizaram bibliotecas de terceiros.

Questão 3: **Conhecimento do sistema que visualizou**.

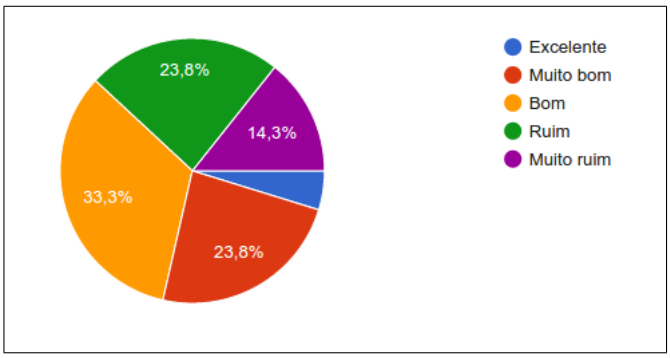

Figura 43 – Resposta da questão 3: Conhecimento do sistema que visualizou.

Fonte: Da autora.

Com as respostas apresentadas na Figura 43, é possível notar que entre os usuários que

responderam ao questionário existem pessoas que conhecem e que não conhecem o sistema visualizado.

6.1.2 Usabilidade

Questões foram aplicadas para que os usuários pudessem avaliar a usabilidade do LibViews.

Questão 4: **Tempo de resposta para a construção da representação**.

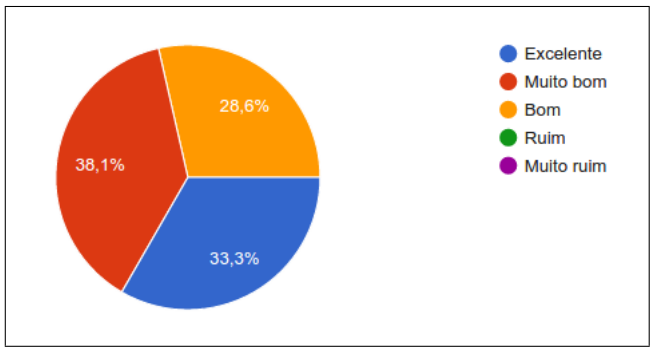

Figura 44 – Resposta da questão 4: Tempo de resposta para a construção da representação.

Fonte: Da autora.

Com as respostas apresentadas na Figura 44, é possível notar que entre os usuários avaliaram entre bom e excelente o tempo de resposta do LibViews quanto a construção da representação.

Questão 5: **Apresentação das informações necessárias para o entendimento da representação**.

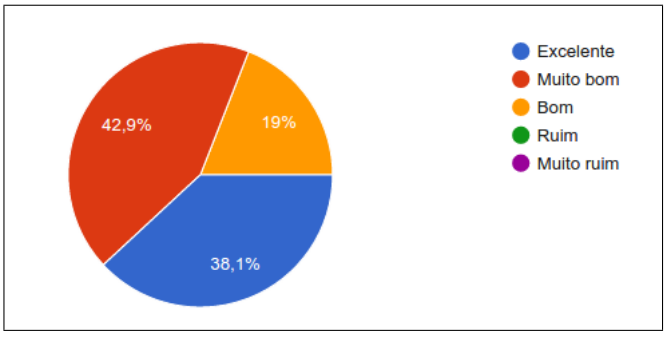

Figura 45 – Resposta da questão 5: Apresentação das informações necessárias para o entendimento da representação.

Fonte: Da autora.

Com as respostas apresentadas na Figura 45, é possível notar que entre os usuários que responderam ao questionário avaliaram entre bom e excelente a apresentação dos dados necessários para o entendimento da representação.

Questão 6: **Disposição das informações e conforto visual**.

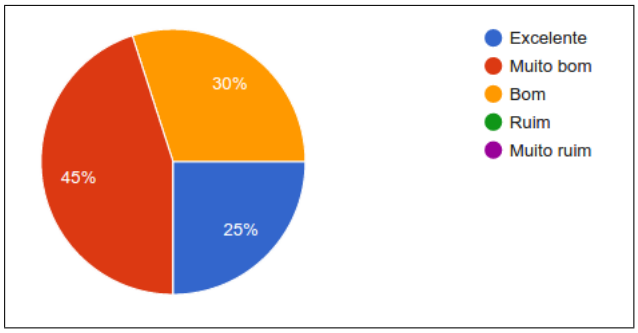

Figura 46 – Resposta da questão 6: Disposição das informações e conforto visual.

Fonte: Da autora.

Com as respostas apresentadas na Figura 46, os usuários aprovaram a disposição das informações e conforto visual do sistema.

6.1.3 Eficiência

Questões foram aplicadas para que os usuários opinassem quanto à eficiência do LibViews.

Questão 7: **Detalhamento das dependências entre bibliotecas e os pacotes que as utilizam**.

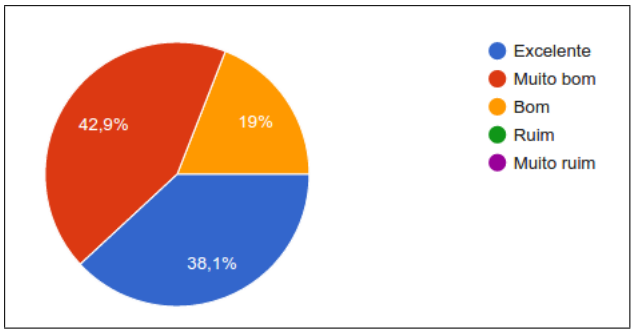

Figura 47 – Resposta da questão 7: Detalhamento das dependências entre bibliotecas e os pacotes que as utilizam.

Fonte: Da autora.

Com as respostas apresentadas na Figura 47, os usuários avaliaram entre bom e excelente o detalhamento entre as dependências entre bibliotecas e o projeto de software.

Questão 8: **Apresentação correta dos dados**.

Com as respostas apresentadas na Figura 48, os usuários validaram como correta a apresentação dos dados.

Questão 9: **Importância da representação completa do seu projeto e suas dependências em uma só tela**.

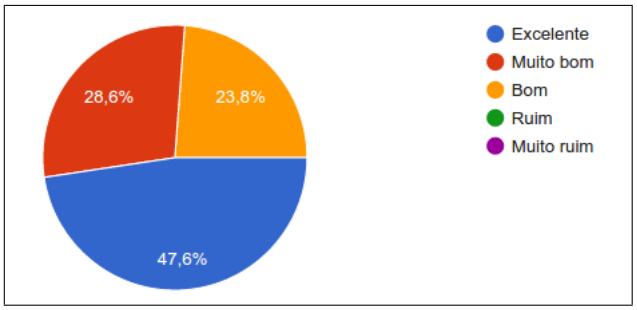

Figura 48 – Resposta da questão 8: Apresentação correta dos dados.

Fonte: Da autora.

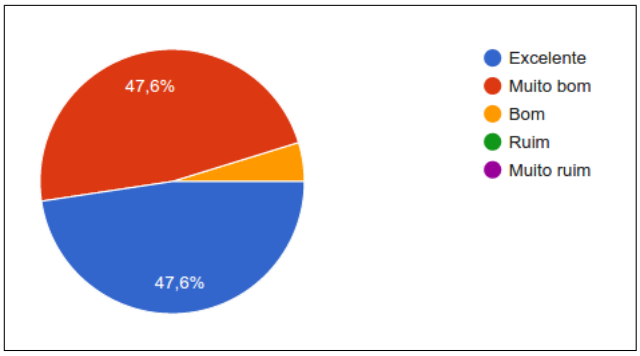

Figura 49 – Resposta da questão 9: Importância da representação completa do seu projeto e suas dependências em uma só tela.

Fonte: Da autora.

Com as respostas apresentadas na Figura 49, os usuários afirmaram a importância do entendimento das informações das dependências em uma só tela.

Questão 10: **Importância do entendimento dos relacionamentos entre os pacotes do projeto e suas bibliotecas**.

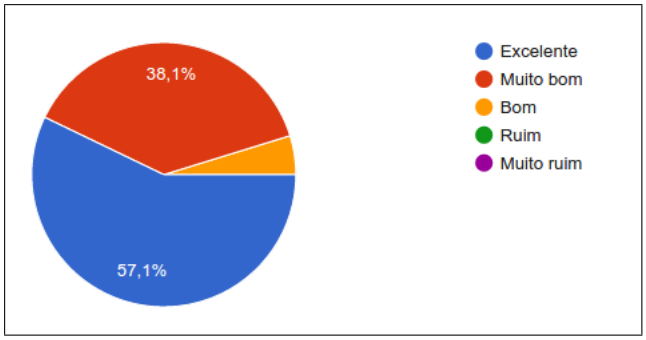

Figura 50 – Resposta da questão 10: Importância do entendimento dos relacionamentos entre os pacotes do projeto e suas bibliotecas.

Fonte: Da autora.

Com as respostas apresentadas na Figura 50, pode-se concluir que é importante o entendimento que o LibViews proporciona.

Questão 11: **Facilidade de uso e entendimento dos recursos do sistema**.

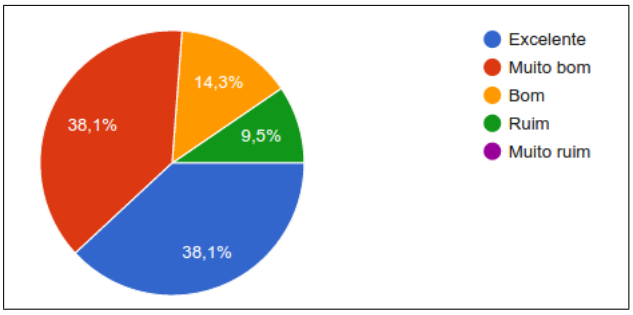

Figura 51 – Resposta da questão 11: Facilidade de uso e entendimento dos recursos do sistema.

Fonte: Da autora.

Com as respostas apresentadas na Figura 51, a maioria dos usuários avaliou bem a facilidade de uso e o entendimento dos recursos do LibViews.

Questão 12: **Descoberta de informações pela representação sobre o projeto e/ou dependências**.

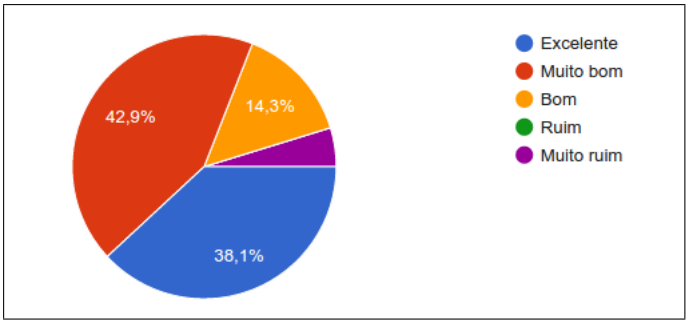

Figura 52 – Resposta da questão 12: Descoberta de informações pela representação sobre o projeto e/ou dependências.

Fonte: Da autora.

Com as respostas apresentadas na Figura 52, a maioria dos usuários afirmaram que a representação permite descoberta de informações.

6.1.4 Satisfação

Questões foram aplicadas para avaliar a satisfação dos usuários ao utilizarem o LibViews.

Questão 13: **Quanto ao entendimento do gráfico que apresenta o número de métodos removidos em cada versão da biblioteca Primefaces**.

Com as respostas apresentadas na Figura 53, a maioria dos usuários entendeu o gráfico apresentado para a métrica NRM.

Questão 14: **Você utilizaria uma biblioteca que tem métodos removidos frequentemente?**

Com as respostas apresentadas na Figura 54, mais de 75% dos usuários não utilizaria uma biblioteca que tem métodos removidos frequentemente, pode-se concluir, portanto, que

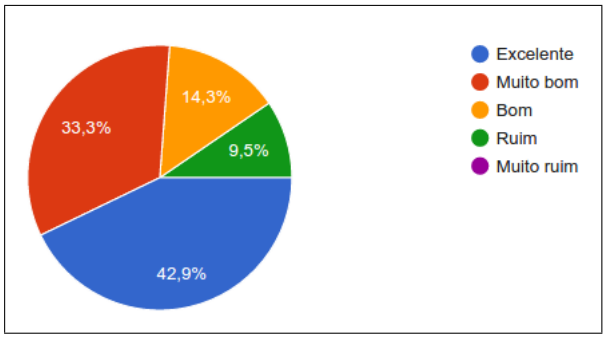

Figura 53 – Resposta da questão 13: Quanto ao entendimento do gráfico que apresenta o número de métodos removidos em cada versão da biblioteca Primefaces.

Fonte: Da autora.

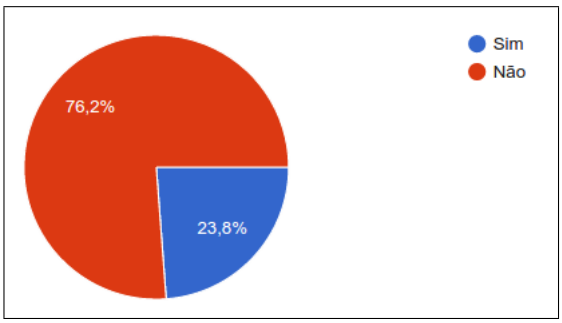

Figura 54 – Resposta da questão 14: Você utilizaria uma biblioteca que tem métodos removidos frequentemente?

Fonte: Da autora.

conhecer o número de métodos removidos é decisivo para definir se a biblioteca será ou não utilizada.

Questão 15: **Quanto ao entendimento do gráfico que apresenta número de métodos de cada versão da biblioteca Primefaces**.

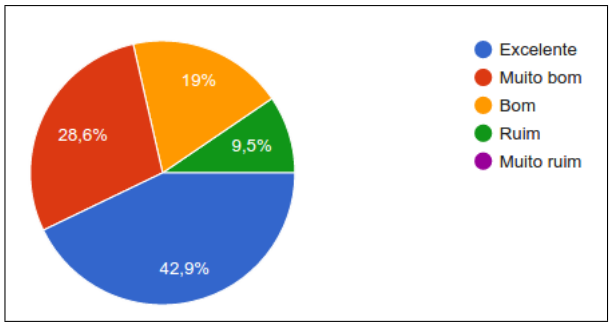

Figura 55 – Resposta da questão 15: Quanto ao entendimento do gráfico que apresenta número de métodos de cada versão da biblioteca Primefaces.

Fonte: Da autora.

Com as respostas apresentadas na Figura 55, a maioria dos usuários entenderam o gráfico que apresenta a métrica NSM.

Questão 16: **Você acha importante saber o tamanho da biblioteca que vai utilizar?**

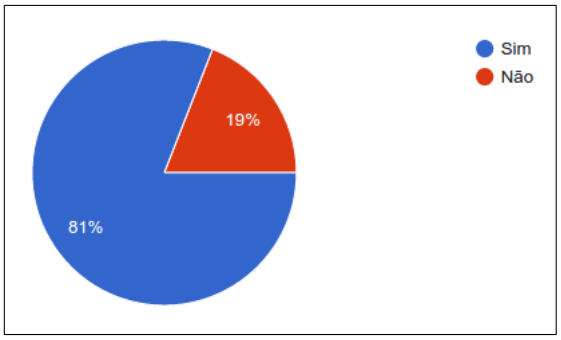

Figura 56 – Resposta da questão 16: Você acha importante saber o tamanho da biblioteca que vai utilizar?

Fonte: Da autora.

Com as respostas apresentadas na Figura 56, mais de 80% dos usuários acha importante saber o tamanho da biblioteca e a métrica NSM fornece esse conhecimento.

Questão 17: **Quanto ao entendimento do gráfico que apresenta número de métodos novos de cada versão da biblioteca Primefaces**.

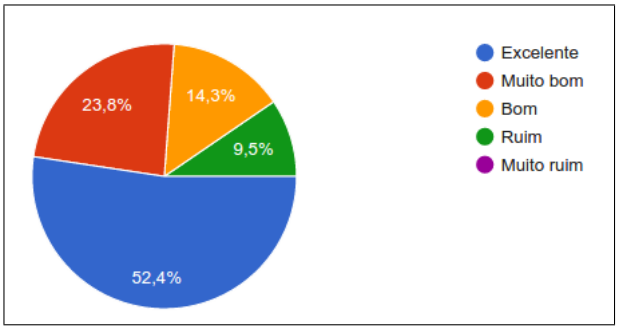

Figura 57 – Resposta da questão 17: Quanto ao entendimento do gráfico que apresenta número de métodos novos de cada versão da biblioteca Primefaces.

Fonte: Da autora.

Com as respostas apresentadas na Figura 57, a maioria dos usuários entendeu o gráfico da métrica NNM.

Questão 18: **Você acha que a biblioteca apresentada no último gráfico está em contínuo desenvolvimento?**

Com as respostas apresentadas na Figura 58, mais de 90% dos usuários entendeu que ao apresentar métodos novos entre as versões a biblioteca está em desenvolvimento.

Questão 19: **Você acha importante saber se a biblioteca está em contínuo desenvolvimento?**

Com as respostas apresentadas na Figura 59, mais de 90% dos usuários acha importante saber se a biblioteca está em contínuo desenvolvimento, o que comprova a importância da métrica NNM.

Questão 20: **Você utilizaria uma biblioteca que há tempos não é atualizada?**

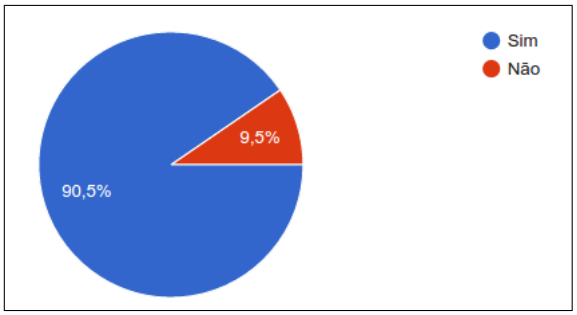

Figura 58 – Resposta da questão 18: Você acha que a biblioteca apresentada no último gráfico está em contínuo desenvolvimento?

Fonte: Da autora.

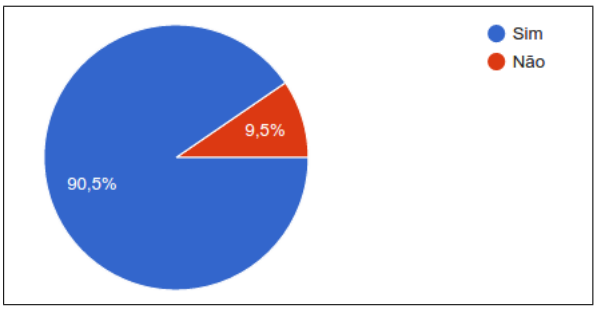

Figura 59 – Resposta da questão 19: Você acha importante saber se a biblioteca está em contínuo desenvolvimento?

Fonte: Da autora.

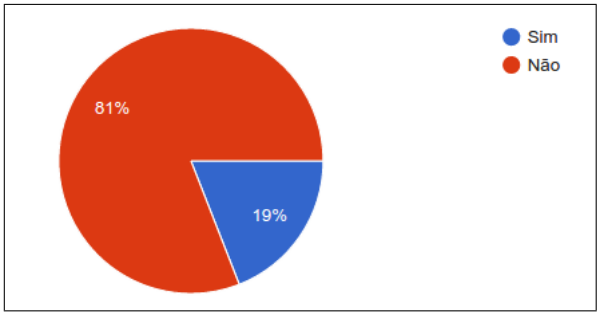

Figura 60 – Resposta da questão 20: Você utilizaria uma biblioteca que há tempos não é atualizada?

Fonte: Da autora.

Com as respostas apresentadas na Figura 60, mais de 80% dos usuários que responderam ao questionário não utilizaria uma biblioteca que não é atualizada há tempos.

Questão 21: **Você utilizaria o LibViews no planejamento de um novo projeto?**

Com as respostas apresentadas na Figura 61, mais de 85% dos usuários que responderam ao questionário utilizariam o LibViews no planejamento de um novo projeto.

Questão 22: **Você utilizaria o LibViews para manutenção de projetos?**

Com as respostas apresentadas na Figura 62, mais de 85% dos usuários que responderam ao questionário utilizaria o LibViews para manutenção de projetos.

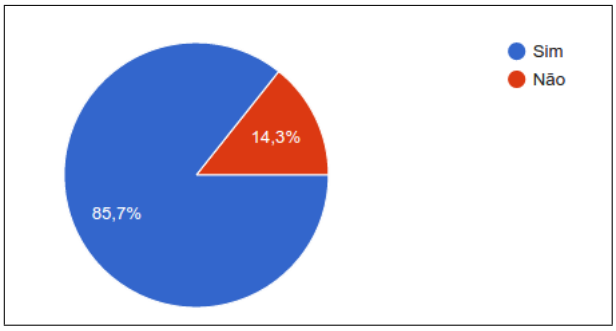

Figura 61 – Resposta da questão 21: Você utilizaria o LibViews no planejamento de um novo projeto?

Fonte: Da autora.

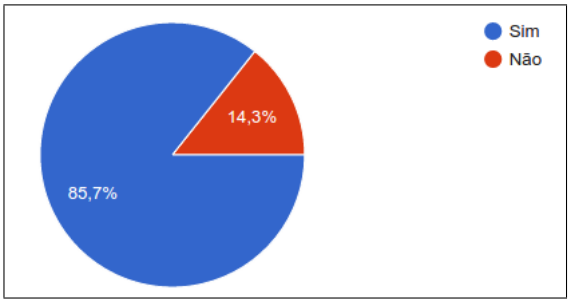

Figura 62 – Resposta da questão 22: Você utilizaria o LibViews para manutenção de projetos?

Fonte: Da autora.

6.2 Considerações finais

Os usuários que responderam ao questionário conheciam a linguagem de programação Java, dentre eles alguns avaliaram o conhecimento com bibliotecas em diferentes níveis, como ruim ou até mesmo excelente. Quanto ao sistema visualizado para responder ao questionário, que é apresentado no caso de uso deste trabalho, alguns usuários o conheciam, outros não. Dessa forma é possível analisar o entendimento que o LibViews proporciona a diferentes níveis de conhecimento dos usuários com relação aos conceitos envolvidos.

A usabilidade do LibViews foi bem avaliada, sendo que praticamente todas as questões foram respondidas com a qualificação "Bom", "Muito bom"ou "Excelente"para as questões dessa área.

Quanto à satisfação do usuário, os resultados também foram positivos: os usuários afirmaram que a representação e o conhecimento das informações que ela disponibiliza são importantes, e que ela gera descoberta de informações que é o intuito da VI; os gráficos, na grande maioria das respostas, foram entendidos; e a importância das métricas foi comprovada, já que as informações que elas fornecem foram confirmadas pelos usuários como decisivas para a escolha das bibliotecas que serão utilizadas.

Finalmente, mais de 85% dos usuários usuários afirmaram que usuariam o LibViews tanto no desenvolvimento de projetos novos, quanto na manutenção de softwares já existentes.

7 Conclusões

O LibViews foi criado a partir de uma necessidade no desenvolvimento de software, comprovada após realização de uma pesquisa acadêmica, que é o gerenciamento de bibliotecas em um projeto de software.

A ferramenta possibilita a apresentação de informações até então desconhecidas, como qual a dependência de cada pacote com relação às bibliotecas, e também apresenta informações dispersas do projeto em uma só tela, como todas as bibliotecas que ele utiliza, ou até, quais bibliotecas são listadas para serem utilizadas e compiladas junto com o projeto, mas não estão em uso.

Seu uso é facilitado por ser acessado em qualquer dispositivo.

E, a partir da avaliação de alguns usuários e também das funções que o LibViews apresenta, pode-se afirmar sua utilidade e o ganho no desenvolvimento de sistemas novos e manutenção de sistemas.

Como trabalhos futuros, melhorias podem ser feitas:

- Apresentação de detalhes nos gráficos, como: quais foram os métodos removidos e quais são os métodos novos; e
- Apresentação das classes que dependem daquela biblioteca, não só os pacotes;
- Visualização de projetos desenvolvidos em Java que não utilizem o Maven;
- Módulo administrativo com sugestões de melhorias no código do projeto.

Para que qualquer desenvolvedor possa fazer uso da ferramenta, ela está disponível no GitHub [1]. No repositório, um arquivo *README.md* fornece as instruções necessárias para a utilização do LibViews.

Esse trabalho foi publicado nos seguintes eventos:

- FERRAREZI, J. C.; BREGA, J. R. F. Visualização de Bibliotecas e suas Dependências em Sistemas de Informação. In: V Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP. Bauru, SP, Brasil: WPPGCC, 2015.
- FERRAREZI, J. C.; BREGA, J. R. F.; POPOLIN NETO, M; DIAS, D. R. C.; PILASTRI, A. L.; GUIMARAES, M. d. P. LibViews - An Information Visualization Application for Third-Party Libraries on Software Projects,"2016 20th International Conference Information Visualisation (IV), Lisbon, 2016, pp. 136-140. doi: 10.1109/IV.2016.43

[1] https://github.com/julianaferrarezi/libviews

O exemplo de utilização e a aplicação apresentados visaram apenas sistemas e bibliotecas desenvolvidos em Java, porém a representação é criada a partir da importação de um arquivo JSON que pode ser gerado a partir de qualquer linguagem de programação. Dessa forma, pode ser feito, por exemplo, um algoritmo para analisar dependências JavaScript e gerar um arquivo JSON que a representação será gerada da mesma forma, o que aumenta os que podem ser favorecidos pela representação apresentada.

Referências

ABNT. *ISO 9241-11: Requisitos Ergonômicos para Trabalho de Escritórios com Computadores Parte 11 - Orientações sobre Usabilidade*. 2002. Associação Brasileira de Normas Técnicas, Brasília. Citado na página 57.

ABNT. *NBRISO/IEC9126-1 Engenharia de software - Qualidade de produto - Parte 1: Modelo de qualidade*. 2003. Associação Brasileira de Normas Técnicas, Brasília. Citado na página 57.

ALHENSHIRI, A.; BLUSTEIN, J. Utilizing visualisation for improving web search effectiveness. In: *Information Society (i-Society), 2010 International Conference on*. [S.l.: s.n.], 2010. p. 43–49. Citado na página 20.

ANDREWS, K. *Information Visualisation - Course Notes*. 2013. Citado 2 vezes nas páginas 21 e 22.

API, T. R. *The Reflection API*. 2015. <https://docs.oracle.com/javase/tutorial/reflect/>. [Online; accessed Julho-2015]. Citado na página 51.

BAUER, V. Facts and fallacies of reuse in practice. In: *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*. [S.l.: s.n.], 2013. p. 431–434. ISSN 1534-5351. Citado na página 16.

BAUER, V.; HEINEMANN, L.; DEISSENBOECK, F. A structured approach to assess third-party library usage. In: *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. [S.l.: s.n.], 2012. p. 483–492. ISSN 1063-6773. Citado na página 16.

CARR, D. A.; UNIVERSITET, L. *Guidelines for Designing Information Visualization Applications*. 1999. Citado na página 20.

D3. *D3*. 2015. <http://d3js.org/>. [Online; accessed Jun-2015]. Citado 2 vezes nas páginas 42 e 43.

D3. *D3*. 2016. <http://d3js.org/>. [Online; accessed Nov-2016]. Citado 4 vezes nas páginas 44, 45, 47 e 48.

DHAND, R. Reducing web page post backs through jquery ajax call in a trust based framework. In: *Computing Sciences (ICCS), 2012 International Conference on*. [S.l.: s.n.], 2012. p. 217–219. Citado na página 18.

DIETRICH, J.; JEZEK, K.; BRADA, P. Broken promises: An empirical study into evolution problems in java programs caused by library upgrades. In: *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*. [S.l.: s.n.], 2014. p. 64–73. Citado 2 vezes nas páginas 16 e 30.

FOUNDATION. *Foundation*. 2015. <http://foundation.zurb.com/>. [Online; accessed Jun-2015]. Citado na página 42.

HIBERNATE. *Hibernate. Everything data.* 2014. <http://hibernate.org/>. [Online; accessed November-2014]. Citado na página 17.

HUMPHREY, A.; MENG, Q.; BERZINS, M.; OLIVEIRA, D. C. B.; RAKAMARIC, Z.; GOPALAKRISHNAN, G. Systematic debugging methods for large-scale hpc computational frameworks. *Computing in Science Engineering*, v. 16, n. 3, p. 48–56, May 2014. ISSN 1521-9615. Citado na página 16.

JING, W.; FAN, R. The research of hibernate cache technique and application of ehcache component. In: *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*. [S.l.: s.n.], 2011. p. 160–162. Citado na página 17.

JSF. *JSF*. 2015. <https://javaserverfaces.java.net/>. [Online; accessed Jun-2015]. Citado na página 42.

KEIL, M.; THIEMANN, P. Type-based dependency analysis for javascript. In: *Proceedings of the Eighth ACM SIGPLAN Workshop on Programming Languages and Analysis for Security*. New York, NY, USA: ACM, 2013. (PLAS '13), p. 47–58. ISBN 978-1-4503-2144-0. Disponível em: <http://doi.acm.org/10.1145/2465106.2465118>. Citado na página 16.

KIRK, A. *Data Visualization: A Sucessful Design Process*. [S.l.]: Packt Publishing Ltd, 2012. Citado 7 vezes nas páginas 21, 22, 23, 24, 25, 26 e 27.

KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J.; LINKMAN, S. Systematic literature reviews in software engineering - a systematic literature review. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 51, n. 1, p. 7–15, jan. 2009. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2008.09.009>. Citado 2 vezes nas páginas 28 e 29.

LAKOFF, G.; JOHNSON, M. The metaphorical structure of the human conceptual system. In: *Cognitive Science*. [S.l.: s.n.], 1980. p. 195–208. Citado na página 22.

LI, J.; LI, H. Management system of sanbao village peace district based on jquery. In: *Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on*. [S.l.: s.n.], 2013. p. 336–339. Citado na página 18.

M2ECLIPSE. *M2Eclipse*. 2014. <http://eclipse.org/m2e/>. [Online; accessed November-2014]. Citado na página 35.

MAJID, E. S. A.; KAMARUDDIN, N.; MANSOR, Z. Adaptation of usability principles in responsive web design technique for e-commerce development. In: *Electrical Engineering and Informatics (ICEEI), 2015 International Conference on*. [S.l.: s.n.], 2015. p. 726–729. Citado na página 41.

MAVEN. *Apache Maven Project*. 2014. <http://maven.apache.org/>. [Online; accessed November-2014]. Citado 3 vezes nas páginas 39, 40 e 41.

MCINTOSH, S.; ADAMS, B.; HASSAN, A. The evolution of ant build systems. In: *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. [S.l.: s.n.], 2010. p. 42–51. Citado na página 39.

MORAES, A.; ELER, D.; BREGA, J. Collaborative information visualization using a multi-projection system and mobile devices. In: *Information Visualisation (IV), 2014 18th International Conference on*. [S.l.: s.n.], 2014. p. 71–77. ISSN 1550-6037. Citado na página 20.

OLIVEIRA, E. Chagas de; CARDOSO, A.; OLIVEIRA, L. Chagas de; LAMOUNIER, E. A proposal for a meta-information visualization using treemap. In: *Computational Science and Computational Intelligence (CSCI), 2014 International Conference on*. [S.l.: s.n.], 2014. v. 1, p. 247–252. Citado na página 44.

ORACLE. *Oracle*. 2015. <http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html>. [Online; accessed Jun-2015]. Citado na página 42.

OSSHER, J.; BAJRACHARYA, S.; LOPES, C. Automated dependency resolution for open source software. In: *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. [S.l.: s.n.], 2010. p. 130–140. Citado na página 39.

PIERRO, A.; CAVALLARI, F.; GUIDA, S. D.; INNOCENTE, V.; BEINARAVICIUS, A. Sql-jquery client a tool managing the db backend of the cms software framework through web browser. In: *Real Time Conference (RT), 2010 17th IEEE-NPSS*. [S.l.: s.n.], 2010. p. 1–4. Citado 2 vezes nas páginas 18 e 19.

PRIMEFACES. *Overview Primefaces 3.5*. 2015. <http://www.primefaces.org/docs/api/3.5/>. [Online; accessed Jun-2015]. Citado na página 52.

PRIMEFACES. *Primefaces*. 2015. <http://www.primefaces.org/>. [Online; accessed Jun-2015]. Citado na página 42.

RAEMAEKERS, S.; DEURSEN, A. van; VISSER, J. Measuring software library stability through historical version analysis. In: *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. [S.l.: s.n.], 2012. p. 378–387. ISSN 1063-6773. Citado 7 vezes nas páginas 32, 33, 34, 35, 40, 42 e 50.

SARAIYA, P.; NORTH, C.; DUCA, K. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE Transactions on Visualization and Computer Graphics*, IEEE Computer Society, Los Alamitos, CA, USA, v. 11, n. 4, p. 443–456, 2005. ISSN 1077-2626. Citado na página 20.

TEYTON, C.; FALLERI, J.-R.; MORANDAT, F.; BLANC, X. Find your library experts. In: *Reverse Engineering (WCRE), 2013 20th Working Conference on*. [S.l.: s.n.], 2013. p. 202–211. Citado na página 16.

W3C. *W3C*. 2015. <http://www.w3c.br/>. [Online; accessed Jun-2015]. Citado 2 vezes nas páginas 41 e 43.

W3SCHOOLS. *W3Schools*. 2016. <http://www.w3schools.com/>. [Online; accessed Nov-2016]. Citado 4 vezes nas páginas 42, 43, 44 e 45.

WARE, C. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2012. (Information Visualization: Perception for Design). ISBN 9780123814647. Disponível em: <https://books.google.com.br/books?id=qFmS95vf6H8C>. Citado 3 vezes nas páginas 20, 21 e 22.

WU, P.; YIN, K. Application research on a persistent technique based on hibernate. In: *Computer Design and Applications (ICCDA), 2010 International Conference on*. [S.l.: s.n.], 2010. v. 1, p. V1–629–V1–631. Citado na página 17.

XUE, M.; ZHU, C. Design and implementation of the hibernate persistence layer data report system based on j2ee. In: *Circuits, Communications and Systems, 2009. PACCS '09. Pacific-Asia Conference on*. [S.l.: s.n.], 2009. p. 232–235. Citado na página 17.

ZHEN-HAI, X.; YONG-ZHI, Y. Automatic updating method based on maven. In: *Computer Science Education (ICCSE), 2014 9th International Conference on*. [S.l.: s.n.], 2014. p. 1074–1077. Citado 2 vezes nas páginas 39 e 41.

APÊNDICE A – Apêndice

A.1 Artigos candidatos da Revisão Sistemática

Tabela 8 – Artigos candidados da revisão sistemática de LB1 - IEEE - Parte 1.

Artigo
Bauer, V.; Heinemann, L., "Understanding API Usage to Support Informed Decision Making in Software Maintenance,"Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on , vol., no., pp.435,440, 27-30 March 2012 doi: 10.1109/CSMR.2012.55
Bauer, V., "Facts and Fallacies of Reuse in Practice,"Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on , vol., no., pp.431,434, 5-8 March 2013 doi: 10.1109/CSMR.2013.65
Bauer, V.; Heinemann, L.; Deissenboeck, F., "A structured approach to assess third-party library usage,"Software Maintenance (ICSM), 2012 28th IEEE International Conference on , vol., no., pp.483,492, 23-28 Sept. 2012 doi: 10.1109/ICSM.2012.6405311
Raemaekers, S.; Nane, G.F.; van Deursen, A.; Visser, J., "Testing principles, current practices, and effects of change localization,"Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on , vol., no., pp.257,266, 18-19 May 2013 doi: 10.1109/MSR.2013.6624037
Grechanik, M.; Chen Fu; Xie, Qing; McMillan, C.; Poshyvanyk, D.; Cumby, C., "A search engine for finding highly relevant applications,"Software Engineering, 2010 ACM/IEEE 32nd International Conference on , vol.1, no., pp.475,484, 2-8 May 2010 doi: 10.1145/1806799.1806868
Pei Wang; Jingiu Yang; Lin Tan; Kroeger, R.; David Morgenthaler, J., "Generating precise dependencies for large software,"Managing Technical Debt (MTD), 2013 4th International Workshop on , vol., no., pp.47,50, 20-20 May 2013 doi: 10.1109/MTD.2013.6608678
Raemaekers, S.; van Deursen, A.; Visser, J., "Measuring software library stability through historical version analysis,"Software Maintenance (ICSM), 2012 28th IEEE International Conference on , vol., no., pp.378,387, 23-28 Sept. 2012 doi: 10.1109/ICSM.2012.6405296
Anslow, C.; Marshall, S.; Noble, J.; Biddle, R., "SourceVis: Collaborative software visualization for co-located environments,"Software Visualization (VISSOFT), 2013 First IEEE Working Conference on , vol., no., pp.1,10, 27-28 Sept. 2013 doi: 10.1109/VISSOFT.2013.6650527
McIntosh, S.; Adams, B.; Hassan, A.E., "The evolution of ANT build systems,"Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on , vol., no., pp.42,51, 2-3 May 2010 doi: 10.1109/MSR.2010.5463341
Teyton, C.; Falleri, J.-R.; Morandat, F.; Blanc, X., "Find your library experts,"Reverse Engineering (WCRE), 2013 20th Working Conference on , vol., no., pp.202,211, 14-17 Oct. 2013 doi: 10.1109/WCRE.2013.6671295
Chengnian Sun; Siau-Cheng Khoo; Shao Jie Zhang, "Graph-based detection of library API imitations,"Software Maintenance (ICSM), 2011 27th IEEE International Conference on , vol., no., pp.183,192, 25-30 Sept. 2011 doi: 10.1109/ICSM.2011.6080785
Dietrich, J.; Jezek, K.; Brada, P., "Broken promises: An empirical study into evolution problems in Java programs caused by library upgrades,"Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on , vol., no., pp.64,73, 3-6 Feb. 2014 doi: 10.1109/CSMR-WCRE.2014.6747226

Tabela 9 – Artigos candidados da revisão sistemática de LB1 - IEEE - Parte 2.

Artigo
Teyton, C.; Falleri, J.-R.; Blanc, X., "Automatic discovery of function mappings between similar libraries,"Reverse Engineering (WCRE), 2013 20th Working Conference on , vol., no., pp.192,201, 14-17 Oct. 2013 doi: 10.1109/WCRE.2013.6671294
Palepu, V.K.; Guoqing Xu; Jones, J.A., "Improving efficiency of dynamic analysis with dynamic dependence summaries,"Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on , vol., no., pp.59,69, 11-15 Nov. 2013 doi: 10.1109/ASE.2013.6693066

Tabela 10 – Artigos candidados da revisão sistemática de LB1 - ACM.

Artigo
BAUER, Veronika et al. An exploratory study on reuse at google. In: Proceedings of the 1st International Workshop on Software Engineering Research and Industrial Practices. New York, NY, USA: ACM, 2014. (SER38;IPs 2014), p. 14–23. ISBN 978-1-4503-2859-3.
CHANG, Jian et al. Tomato: A trustworthy code mashup development tool. In: Proceedings of the 5th International Workshop on Web APIs and Service Mashups. New York, NY, USA: ACM, 2011. (Mashups '11), p. 5:1–5:8. ISBN 978-1-4503-0823-6.
GRECHANIK, Mark et al. A search engine for finding highly relevant applications. In: Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1. New York, NY, USA: ACM, 2010. (ICSE '10), p. 475–484. ISBN 978-1-60558- 719-6.
JANSEN, Slinger. How quality attributes of software platform architectures influence soft- ware ecosystems. In: Proceedings of the 2013 International Workshop on Ecosystem Archi- tectures. New York, NY, USA: ACM, 2013. (WEA 2013), p. 6–10. ISBN 978-1-4503-2314-7.
KEIL, Matthias; THIEMANN, Peter. Type-based dependency analysis for javascript. In: Proceedings of the Eighth ACM SIGPLAN Workshop on Programming Languages and Analysis for Security. New York, NY, USA: ACM, 2013. (PLAS '13), p. 47–58. ISBN 978-1-4503- 2144-0.
LINARES-VáSQUEZ, Mario et al. Revisiting android reuse studies in the context of code obfuscation and library usages. In: Proceedings of the 11th Working Conference on Mining Software Repositories. New York, NY, USA: ACM, 2014. (MSR 2014), p. 242–251. ISBN 978-1-4503-2863-0.
RAEMAEKERS, Steven et al. Testing principles, current practices, and effects of change localization. In: Proceedings of the 10th Working Conference on Mining Software Repositories. Piscataway, NJ, USA: IEEE Press, 2013. (MSR '13), p. 257–266. ISBN 978-1-4673-2936-1.
SCHWITTEK, Widura; EICKER, Stefan. A study on third party component reuse in java enterprise open source software. In: Proceedings of the 16th International ACM Sigsoft Symposium on Component-based Software Engineering. New York, NY, USA: ACM, 2013. (CBSE '13), p. 75–80. ISBN 978-1-4503-2122-8.
SONG, Myoungkyu; TILEVICH, Eli. Tae-js: Automated enhancement of javascript programs by leveraging the java annotations infrastructure. In: Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools. New York, NY, USA: ACM, 2013. (PPPJ '13), p. 13–24. ISBN 978-1-4503-2111-2.
SUN, Mengtao; TAN, Gang. Nativeguard: Protecting android applications from third-party native libraries. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless 38; Mobile Networks. New York, NY, USA: ACM, 2014. (WiSec '14), p. 165–176. ISBN 978-1-4503-2972-9.

Tabela 11 – Artigos candidados da revisão sistemática de LB2 - IEEE - Parte 1.

Artigo
Lawrence-Fowler, W.A.; Grabowski, L.; Fowler, R.H.; Yedid, G., "Convergence of evolutionary biology and software engineering: Putting practice in action,"Frontiers in Education Conference, 2013 IEEE , vol., no., pp.356,361, 23-26 Oct. 2013
Yuan Chen; Xiang-heng Shen; Peng Du; Bing Ge, "Research on software defect prediction based on data mining,"Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on , vol.1, no., pp.563,567, 26-28 Feb. 2010
Abuthawabeh, A.; Beck, F.; Zeckzer, D.; Diehl, S., "Finding structures in multi-type code couplings with node-link and matrix visualizations,"Software Visualization (VISSOFT), 2013 First IEEE Working Conference on , vol., no., pp.1,10, 27-28 Sept. 2013
Xiaofan Chen; Hosking, J.; Grundy, J., "Visualizing traceability links between source code and documentation,"Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium on , vol., no., pp.119,126, Sept. 30 2012-Oct. 4 2012
2012 International Conference on Devices, Circuits and Systems (ICDCS),"Devices, Circuits and Systems (ICDCS), 2012 International Conference on , vol., no., pp.1,748, 15-16 March 2012
Rhyne, T.; Min Chen, "Cutting-Edge Research in Visualization,"Computer , vol.46, no.5, pp.22,24, May 2013
Jusufi, I.; Yang Dingjie; Kerren, A., "The Network Lens: Interactive Exploration of Multivariate Networks Using Visual Filtering,"Information Visualisation (IV), 2010 14th International Conference , vol., no., pp.35,42, 26-29 July 2010
Shaheed, A.; Fergus, P.; Abuelma'atti, O.E.; Merabti, M., "Visualisation of Ubiquitous Home Devices and Services,"Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE , vol., no., pp.1,5, 9-12 Jan. 2010
de Jesus Nascimento da Silva Junior, J.; Meiguins, B.S.; Carneiro, N.S.; Meiguins, A.S.G.; da Silva Franco, R.Y.; Soares, A.G.M., "PRISMA Mobile: An Information Visualization Tool for Tablets,"Information Visualisation (IV), 2012 16th International Conference on , vol., no., pp.182,187, 11-13 July 2012
Vidhu Bhala, R.V.; Mala, T.; Abirami, S., "Effective visualization of conceptual class diagrams,"Recent Advances in Computing and Software Systems (RACSS), 2012 International Conference on , vol., no., pp.1,6, 25-27 April 2012
Kotval, Xerxes P.; Burns, Michael J., "Visualization of entities within social media: Toward understanding users' needs,"Bell Labs Technical Journal , vol.17, no.4, pp.77,102, March 2013
Yunsong Yang, "Three-dimensional visualization technology using in the water dispatching decision support system,"Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on , vol., no., pp.2664,2667, 8-10 Aug. 2011
Zhang Yinan; He Jing; Yuan Jie, "A stereoscopic display system based on a volumetric image model,"Natural Computation (ICNC), 2013 Ninth International Conference on , vol., no., pp.1273,1277, 23-25 July 2013
Sfayhi, A.; Sahraoui, H., "What You See is What You Asked for: An Effort-Based Transformation of Code Analysis Tasks into Interactive Visualization Scenarios,"Source Code Analysis and Manipulation (SCAM), 2011 11th IEEE International Working Conference on , vol., no., pp.195,203, 25-26 Sept. 2011
Duffy, B.; Thiyagalingam, J.; Walton, S.; Smith, D.J.; Trefethen, A.; Kirkman-Brown, J.C.; Gaffney, E.A.; Chen, M., "Glyph-Based Video Visualization for Semen Analysis,"Visualization and Computer Graphics, IEEE Transactions on , vol.21, no.8, pp.980,993, Aug. 2015

Tabela 12 – Artigos candidados da revisão sistemática de LB2 - IEEE - Parte 2.

Artigo
ZhenMin Peng; Grundy, E.; Laramee, R.S.; Guoning Chen; Croft, N., "Mesh-Driven Vector Field Clustering and Visualization: An Image-Based Approach,"Visualization and Computer Graphics, IEEE Transactions on , vol.18, no.2, pp.283,298, Feb. 2012
Alazab, M.; Moonsamy, V.; Batten, L.; Lantz, P.; Ronghua Tian, "Analysis of malicious and benign android applications,"Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on , vol., no., pp.608,616, 18-21 June 2012

Tabela 13 – Artigos candidados da revisão sistemática de LB2 - ACM - Parte 1.

Artigo
Matti Anttonen, Arto Salminen, Tommi Mikkonen, and Antero Taivalsaari. 2011. Transforming the web into a real application platform: new technologies, emerging trends and missing pieces. In Proceedings of the 2011 ACM Symposium on Applied Computing (SAC '11). ACM, New York, NY, USA, 800-807. DOI=10.1145/1982185.1982357 http://doi.acm.org/10.1145/1982185.1982357
John Stasko. 2014. Value-driven evaluation of visualizations. In Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization (BELIV '14), Heidi Lam, Petra Isenberg, Tobias Isenberg, and Michael Sedlmair (Eds.). ACM, New York, NY, USA, 46-53. DOI=10.1145/2669557.2669579 http://doi.acm.org/10.1145/2669557.2669579
Colin Atkinson and Ralph Gerbig. 2013. Harmonizing textual and graphical visualizations of domain specific models. In Proceedings of the Second Workshop on Graphical Modeling Language Development (GMLD '13), Heiko Kern, Juha-Pekka Tolvanen, and Paolo Bottoni (Eds.). ACM, New York, NY, USA, 32-41. DOI=10.1145/2489820.2489823 http://doi.acm.org/10.1145/2489820.2489823
Tommi Mikkonen and Antero Taivalsaari. 2010. The mashware challenge: bridging the gap between web development and software engineering. In Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER '10). ACM, New York, NY, USA, 245-250. DOI=10.1145/1882362.1882413 http://doi.acm.org/10.1145/1882362.1882413
Colin Atkinson, Ralph Gerbig, and Christian Tunjic. 2013. A multi-level modeling environment for SUM-based software engineering. In Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling (VAO '13). ACM, New York, NY, USA, , Article 2 , 9 pages. DOI=10.1145/2489861.2489868 http://doi.acm.org/10.1145/2489861.2489868
Rainer Lutz and Stephan Diehl. 2014. Using visual dataflow programming for interactive model comparison. In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (ASE '14). ACM, New York, NY, USA, 653-664. DOI=10.1145/2642937.2642984 http://doi.acm.org/10.1145/2642937.2642984
2014. Proceedings of the 1st First Workshop for High Performance Technical Computing in Dynamic Languages. IEEE Press, Piscataway, NJ, USA.
Niklas Elmqvist and Ji Soo Yi. 2012. Patterns for visualization evaluation. In Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors - Novel Evaluation Methods for Visualization (BELIV '12). ACM, New York, NY, USA, , Article 12 , 8 pages. DOI=10.1145/2442576.2442588 http://doi.acm.org/10.1145/2442576.2442588

Tabela 14 – Artigos candidados da revisão sistemática de LB2 - ACM - Parte 2.

Artigo
Xiongfei Luo, Dongxing Teng, Wei Liu, Feng Tian, Guozhong Dai, and Hongan Wang. 2010. A developing framework for interactive temporal data visualization. In Proceedings of the 3rd International Symposium on Visual Information Communication (VINCI '10). ACM, New York, NY, USA, , Article 14 , 10 pages. DOI=10.1145/1865841.1865860 http://doi.acm.org/10.1145/1865841.1865860

A.2 Questionário

O LibViews é uma ferramenta para visualização de pacotes de determinado projeto e suas dependências/bibliotecas. Para conhecê-la e responder o questionário, acesse o link: http://slstr.bauru.unesp.br/libviews/administrativo.xhtml.

1. Conhecimento da linguagem de programação Java.
    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim
2. Experiência com o uso de bibliotecas.
    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim
3. Conhecimento do sistema que visualizou.
    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim
4. Tempo de resposta para a construção da representação.
    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim
5. Apresentação das informações necessárias para o entendimento da representação.

- Excelente
- Muito bom
- Bom
- Ruim
- Muito ruim

6. Disposição das informações e conforto visual.
    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim
7. Detalhamento das dependências entre bibliotecas e os pacotes que as utilizam.
    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim
8. Apresentação correta dos dados.
    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim
9. Importância da representação completa do seu projeto e suas dependências em uma só tela.
    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim

10. Importância do entendimento dos relacionamentos entre os pacotes do projeto e suas bibliotecas.

    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim

11. Facilidade de uso e entendimento dos recursos do sistema.

    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim

12. Descoberta de informações pela representação sobre o projeto e/ou dependências.

    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim

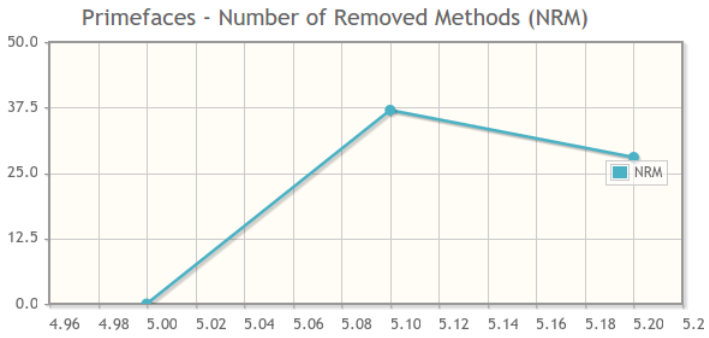

Figura 63 – Métrica NRM para as versões 5.0, 5.1 e 5.2 da biblioteca Primefaces.

Fonte: Da autora.

13. Quanto ao entendimento do gráfico que apresenta o número de métodos removidos em cada versão da biblioteca Primefaces.

    - Excelente

- Muito bom
- Bom
- Ruim
- Muito ruim

14. Você utilizaria uma biblioteca que tem métodos removidos frequentemente?

    - Sim
    - Não

15. Quanto ao entendimento do gráfico que apresenta número de métodos de cada versão da biblioteca Primefaces.

    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim

16. Você acha importante saber o tamanho da biblioteca que vai utilizar?

    - Sim
    - Não

17. Quanto ao entendimento do gráfico que apresenta número de métodos novos de cada versão da biblioteca Primefaces.

    - Excelente
    - Muito bom
    - Bom
    - Ruim
    - Muito ruim

18. Você acha que a biblioteca apresentada no último gráfico está em contínuo desenvolvimento?

    - Sim
    - Não

19. Você acha importante saber se a biblioteca está em contínuo desenvolvimento?

    - Sim

- Não

20. Você utilizaria uma biblioteca que há tempos não é atualizada?

    - Sim
    - Não

21. Você utilizaria o LibViews no planejamento de um novo projeto?

    - Sim
    - Não

22. Você utilizaria o LibViews para manutenção de projetos?

    - Sim
    - Não

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

TERMO DE REPRODUÇÃO XEROGRÁFICA

Autorizo a reprodução xerográfica do presente Trabalho de Conclusão, na íntegra ou em partes, para fins de pesquisa.

São José do Rio Preto, 30 / 01 / 2017

Juliana C. Ferraz

Assinatura do autor